

# Lecture 7

# Aspect-orientation (AO\*)

A new paradigm in Software  
Engineering

Copyright © Yijun Yu, 2005

Last lecture and tutorial ...

## Software Quality Measurements

- We have shown the use of quality measurements to monitor the progress of software development
- The development/restructuring (maintenance) activities (refactoring, tuning, adding features) can be guided by the metrics of softgoals

Today ...

# On Aspect Orientation

- Today we explain the paradigm of aspect-orientation
  1. Concepts: What are aspects?
  2. Practices: Aspect-orientation at large
    - AOP: Aspect-oriented programming
    - AOSD: Aspect-oriented software development
    - *AORE: Aspect-oriented requirements engineering*
    - AOSR: Aspect-oriented software reuse (probably next lecture)
  3. A case study of AORE
  4. Summary

# 1. What are aspects?

1. Some design principles
  - Divide and conquer: problem solving/design principle
  - Modularization: high cohesion/low coupling  
Separation of concerns
  - DRY: Don't Repeat Yourself  
Increase the fan-in
2. Previous paradigms
  - 70s – 80s:  
Structured programming (Goto's considered harmful) =>  
Structured Analysis, Structured Design
  - 80s – 90s:  
Object-oriented programming (OOP) =>  
OOA/OOD => UML
3. Why another paradigm ?
  - Since late 90s ...  
Separation of the *crosscutting* concerns
4. What are aspects?
  - Modularizing the crosscutting concerns

## 1.1 Some design principles

# Structured programming

- What is structured program?
  - A program has no more GOTO's
  - Only three kinds of structure prevails
    - Sequential
    - If-then-else
    - Loops

*[Dijkstra: Goto considered harmful]*

- In other words, every statement block has single-entry, single-exit as Hammock Graph

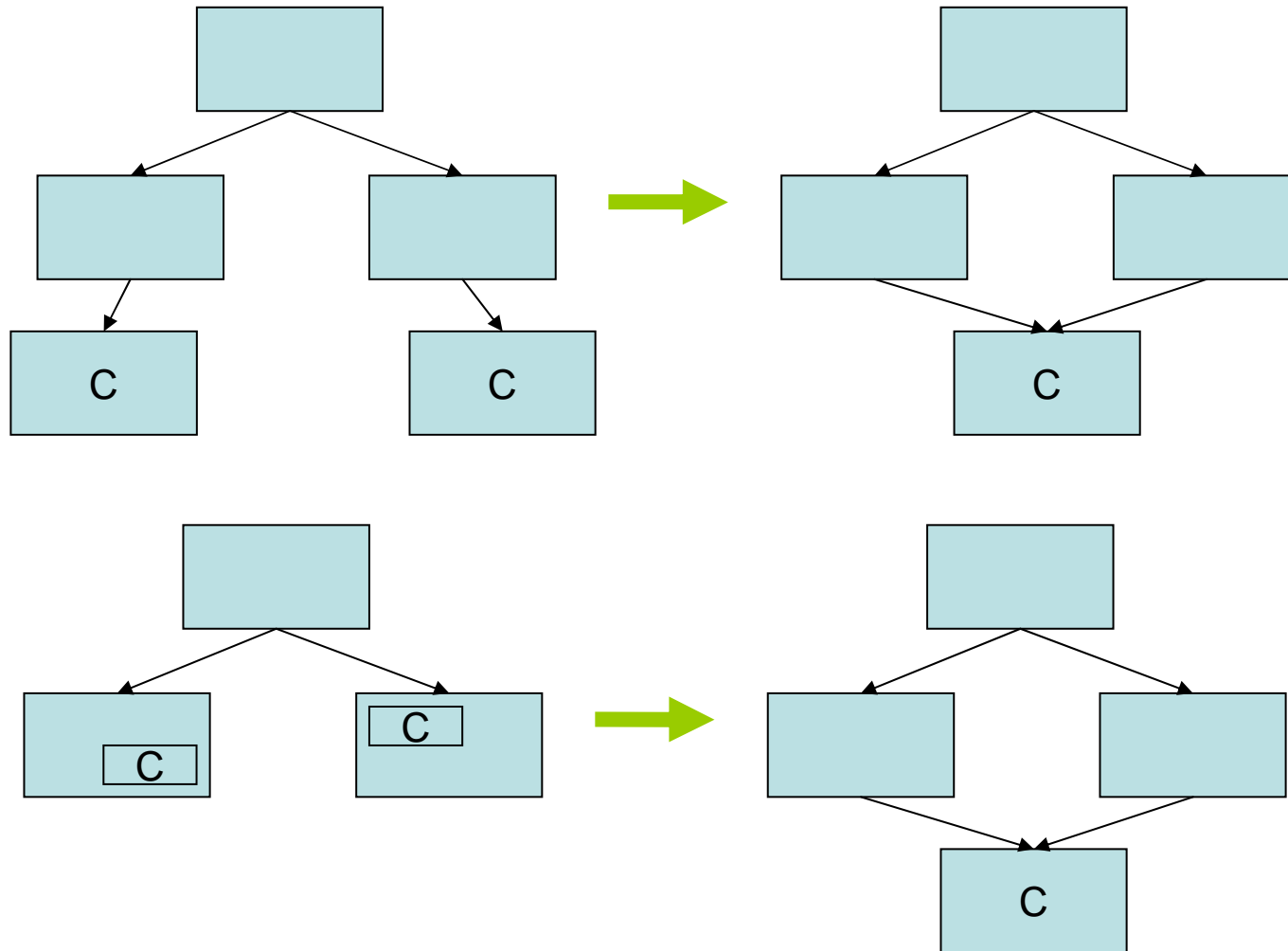
*[Weiser: Program slicing]*
- “Whenever possible, we wish to maximize **fan-in** during the design process. Fan-in is the *raison d'être* of modularity. Each instance of multiple fan-in means that some duplicate code has been avoided.”

**raison d'être:** grounds for existence  
(<http://www.french-linguistics.co.uk/dictionary/>)

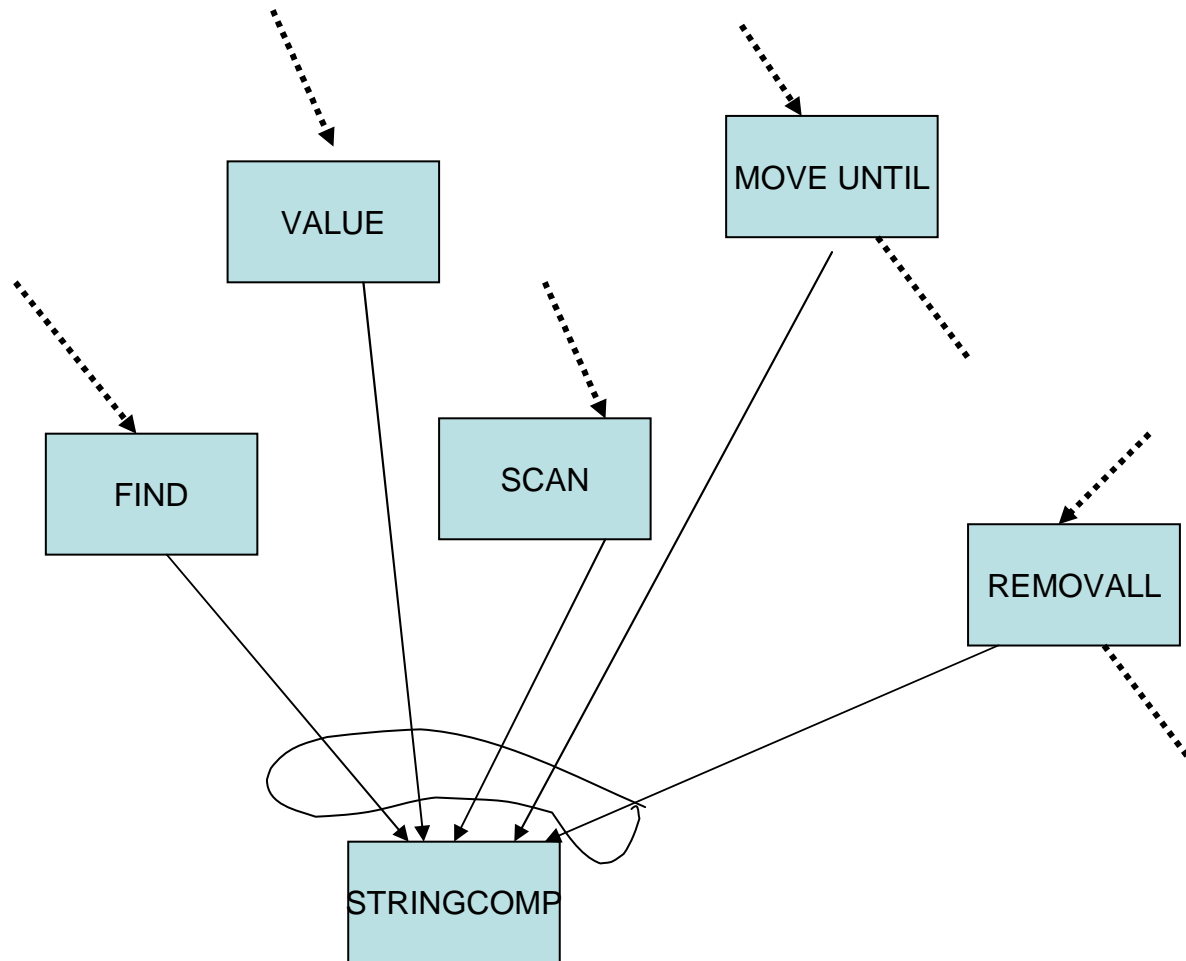
*[Yourdon & Constantine79] Structured Design (pg. 172, see also <http://www.wpa.win.tue.nl/wstomv/quotes/structured-design.html>)*

*[parnas: Modularization, information hiding]*

- (1) A decomposition hierarchy from abstract to concrete:  
Divide and Conquer, Structured Design;  
(2) Don't Repeat Yourself, Factoring / Refactoring ...



# Example



Yourdon & Constantine, SD, pg.168

## 1.1 Some design principles

# Object-oriented programming

- Everything is an object (Smalltalk)
- Information hiding / Encapsulation: object groups related data and the operations on the data into a module
- Object has structural relationships:
  - inheritance: generalization / specialization: isA/instanceOf
  - aggregation : hasA / isPartOf
  - associations: 1-to-many, 1-to-1, many-to-many
- In the end, the structurally-related objects are *packaged* into components



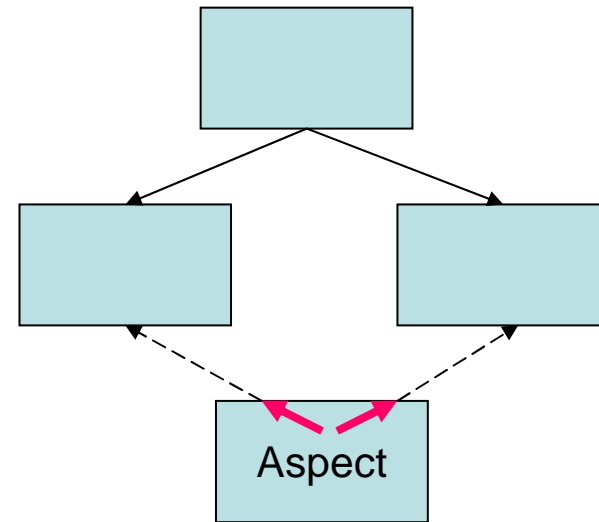
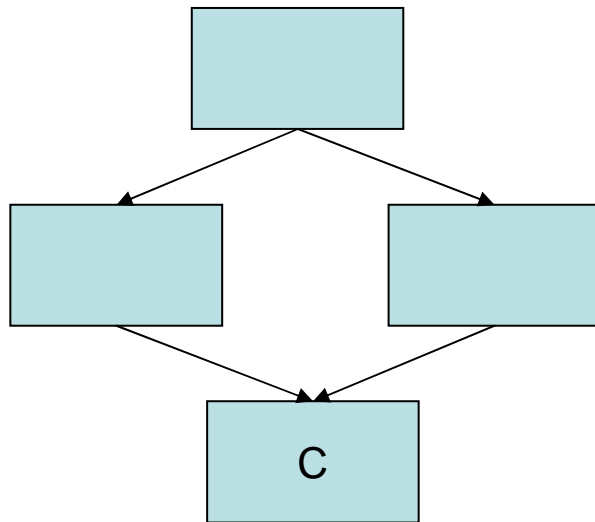
## 1.2 Aspect-orientation

- Component language  
(any structured or OO language, even corresponding design and requirements specification)
- What are crosscutting concerns?
- An aspect language
  - What are joinpoints?
  - What are pointcuts?
  - What are advices?
- A weaving mechanism

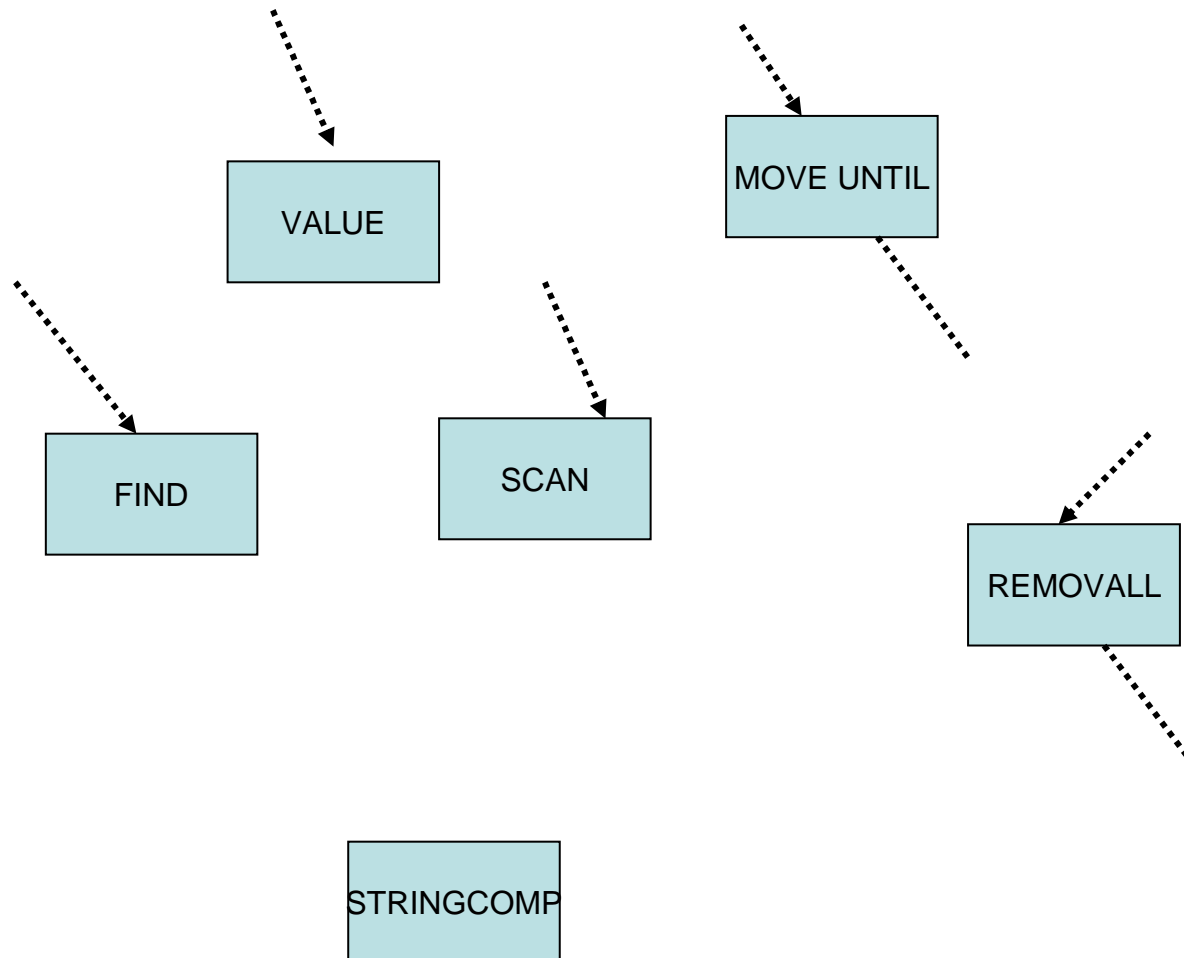
# Aspect concepts

- Concepts:  
cross-cutting,  
component, aspect,  
join points, weaving

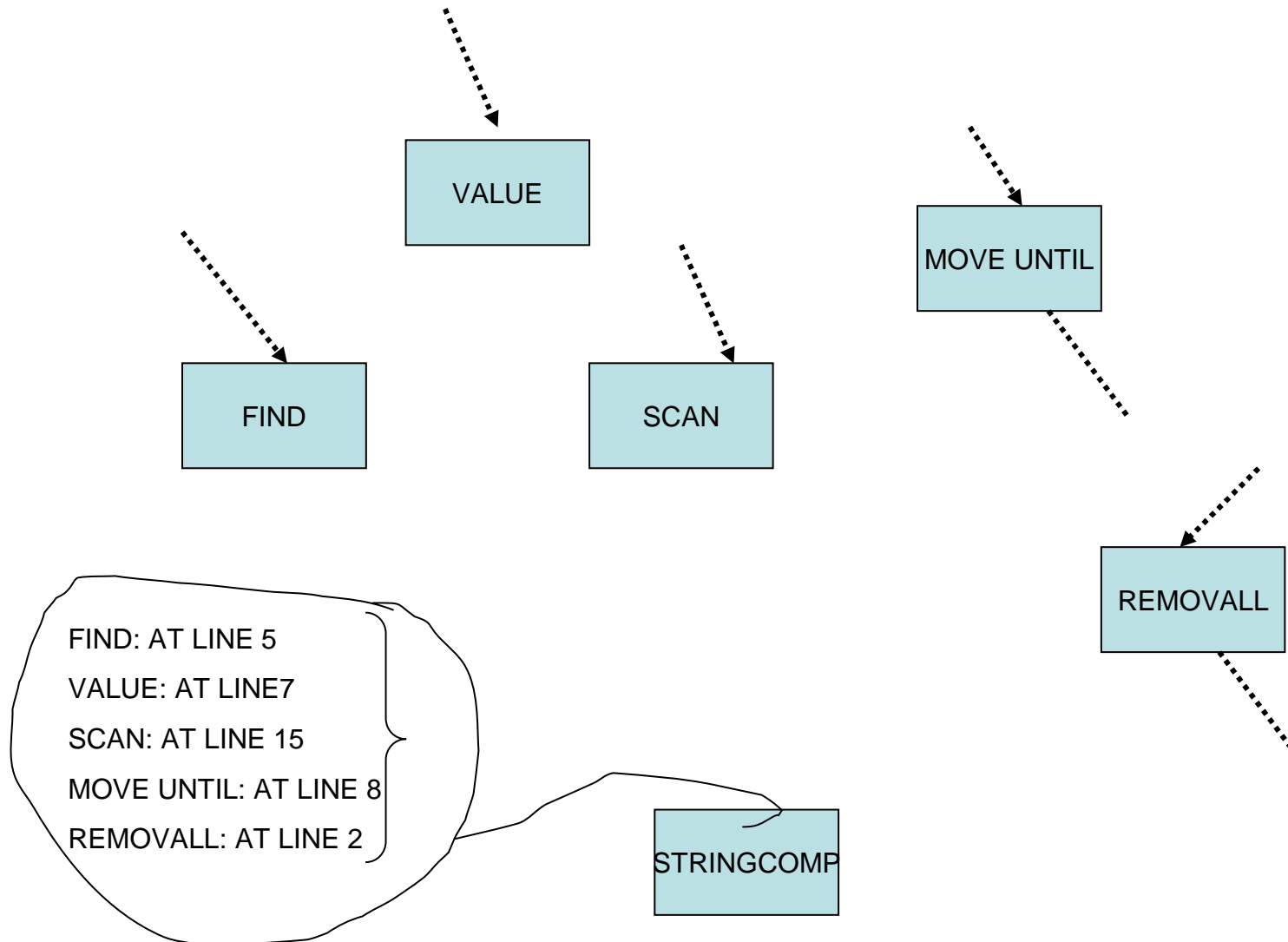
***AOP hides the join points***



# AOP (THE MAGIC)



# AOP (NOT REALLY MAGIC)



# AOP example

## ApplicationSession

[illegible]

## StandardSession

<p>1. The first step in the process of creating a new product is to identify a market need. This involves conducting market research to understand the needs and preferences of potential customers.</p> <p>2. Once a market need is identified, the next step is to develop a product concept. This involves brainstorming ideas and selecting the most promising one.</p> <p>3. The third step is to create a detailed product specification. This document outlines the features, functions, and performance requirements of the product.</p> <p>4. The fourth step is to design the product. This involves creating technical drawings and prototypes to visualize the product and test its feasibility.</p> <p>5. The fifth step is to manufacture the product. This involves sourcing materials, setting up production equipment, and assembling the product.</p> <p>6. The sixth step is to distribute the product. This involves finding distribution channels and getting the product into the hands of customers.</p> <p>7. The seventh step is to promote the product. This involves developing a marketing strategy and implementing promotional activities to create awareness and drive sales.</p> <p>8. The eighth step is to provide customer support. This involves offering assistance to customers who have questions or issues with the product.</p> <p>9. The ninth step is to monitor the product's performance. This involves tracking sales, customer feedback, and market trends to assess the product's success and identify areas for improvement.</p> <p>10. The tenth step is to iterate and improve the product. This involves making changes to the product based on customer feedback and market trends to enhance its value and competitiveness.</p>	<p>1. The first step in the process of creating a new product is to identify a market need. This involves conducting market research to understand the needs and preferences of potential customers.</p> <p>2. Once a market need is identified, the next step is to develop a product concept. This involves brainstorming ideas and selecting the most promising one.</p> <p>3. The third step is to create a detailed product specification. This document outlines the features, functions, and performance requirements of the product.</p> <p>4. The fourth step is to design the product. This involves creating technical drawings and prototypes to visualize the product and test its feasibility.</p> <p>5. The fifth step is to manufacture the product. This involves sourcing materials, setting up production equipment, and assembling the product.</p> <p>6. The sixth step is to distribute the product. This involves finding distribution channels and getting the product into the hands of customers.</p> <p>7. The seventh step is to promote the product. This involves developing a marketing strategy and implementing promotional activities to create awareness and drive sales.</p> <p>8. The eighth step is to provide customer support. This involves offering assistance to customers who have questions or issues with the product.</p> <p>9. The ninth step is to monitor the product's performance. This involves tracking sales, customer feedback, and market trends to assess the product's success and identify areas for improvement.</p> <p>10. The tenth step is to iterate and improve the product. This involves making changes to the product based on customer feedback and market trends to enhance its value and competitiveness.</p>	<p>1. The first step in the process of creating a new product is to identify a market need. This involves conducting market research to understand the needs and preferences of potential customers.</p> <p>2. Once a market need is identified, the next step is to develop a product concept. This involves brainstorming ideas and selecting the most promising one.</p> <p>3. The third step is to create a detailed product specification. This document outlines the features, functions, and performance requirements of the product.</p> <p>4. The fourth step is to design the product. This involves creating technical drawings and prototypes to visualize the product and test its feasibility.</p> <p>5. The fifth step is to manufacture the product. This involves sourcing materials, setting up production equipment, and assembling the product.</p> <p>6. The sixth step is to distribute the product. This involves finding distribution channels and getting the product into the hands of customers.</p> <p>7. The seventh step is to promote the product. This involves developing a marketing strategy and implementing promotional activities to create awareness and drive sales.</p> <p>8. The eighth step is to provide customer support. This involves offering assistance to customers who have questions or issues with the product.</p> <p>9. The ninth step is to monitor the product's performance. This involves tracking sales, customer feedback, and market trends to assess the product's success and identify areas for improvement.</p> <p>10. The tenth step is to iterate and improve the product. This involves making changes to the product based on customer feedback and market trends to enhance its value and competitiveness.</p>
---	---	---

## SessionInterceptor

```

1  # Import the required modules (numpy, pandas, sklearn)
2  import numpy as np
3  import pandas as pd
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
7
8  # Load the dataset
9  data = pd.read_csv('data.csv')
10
11 # Split the data into training and testing sets
12 X_train, X_test, y_train, y_test = train_test_split(
13     data[['feature1', 'feature2', 'feature3', 'feature4', 'feature5', 'feature6', 'feature7', 'feature8', 'feature9', 'feature10']],
14     data['target'],
15     test_size=0.2,
16     random_state=42)
17
18 # Standardize the features
19 scaler = StandardScaler()
20 X_train = scaler.fit_transform(X_train)
21 X_test = scaler.transform(X_test)
22
23 # Train the model
24 # Here, we use a placeholder for the model training process.
25 # The actual model and training code would go here.
26
27 # Evaluate the model
28 y_pred = model.predict(X_test)
29 accuracy = accuracy_score(y_test, y_pred)
30 conf_matrix = confusion_matrix(y_test, y_pred)
31 class_report = classification_report(y_test, y_pred)
32
33 # Print the results
34 print(f'Accuracy: {accuracy}')
35 print('Confusion Matrix:')
36 print(conf_matrix)
37 print('Classification Report:')
38 print(class_report)
39
40 # End of the script

```

## StandardManager

```

1  # Import the random module
2  import random
3
4  # Create a list of 10 random integers between 1 and 100
5  random_list = [random.randint(1, 100) for _ in range(10)]
6
7  # Print the list
8  print(random_list)
9
10 # Create a dictionary with 5 random key-value pairs
11 # The keys are strings and the values are integers
12 random_dict = {}
13 for _ in range(5):
14     key = 'key_' + str(random.randint(0, 999))
15     value = random.randint(1, 100)
16     random_dict[key] = value
17
18 # Print the dictionary
19 print(random_dict)
20
21 # Create a set of 10 random integers between 1 and 100
22 random_set = set(random.randint(1, 100) for _ in range(10))
23
24 # Print the set
25 print(random_set)
26
27 # Create a tuple of 10 random integers between 1 and 100
28 random_tuple = tuple(random.randint(1, 100) for _ in range(10))
29
30 # Print the tuple
31 print(random_tuple)
32
33 # Create a list of 10 random floats between 0.1 and 0.9
34 random_floats = [random.uniform(0.1, 0.9) for _ in range(10)]
35
36 # Print the list
37 print(random_floats)
38
39 # Create a dictionary with 5 random key-value pairs
40 # The keys are strings and the values are floats
41 random_float_dict = {}
42 for _ in range(5):
43     key = 'key_' + str(random.randint(0, 999))
44     value = random.uniform(0.1, 0.9)
45     random_float_dict[key] = value
46
47 # Print the dictionary
48 print(random_float_dict)
49
50 # Create a set of 10 random floats between 0.1 and 0.9
51 random_float_set = set(random.uniform(0.1, 0.9) for _ in range(10))
52
53 # Print the set
54 print(random_float_set)
55
56 # Create a tuple of 10 random floats between 0.1 and 0.9
57 random_float_tuple = tuple(random.uniform(0.1, 0.9) for _ in range(10))
58
59 # Print the tuple
60 print(random_float_tuple)
61
62 # Create a list of 10 random strings of length 5
63 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
64
65 # Print the list
66 print(random_strings)
67
68 # Create a dictionary with 5 random key-value pairs
69 # The keys are strings and the values are strings
70 random_string_dict = {}
71 for _ in range(5):
72     key = 'key_' + str(random.randint(0, 999))
73     value = ''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5))
74     random_string_dict[key] = value
75
76 # Print the dictionary
77 print(random_string_dict)
78
79 # Create a set of 10 random strings of length 5
80 random_string_set = set(''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10))
81
82 # Print the set
83 print(random_string_set)
84
85 # Create a tuple of 10 random strings of length 5
86 random_string_tuple = tuple(''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10))
87
88 # Print the tuple
89 print(random_string_tuple)
90
91 # Create a list of 10 random booleans
92 random_booleans = [random.choice([True, False]) for _ in range(10)]
93
94 # Print the list
95 print(random_booleans)
96
97 # Create a dictionary with 5 random key-value pairs
98 # The keys are strings and the values are booleans
99 random_boolean_dict = {}
100 for _ in range(5):
101     key = 'key_' + str(random.randint(0, 999))
102     value = random.choice([True, False])
103     random_boolean_dict[key] = value
104
105 # Print the dictionary
106 print(random_boolean_dict)
107
108 # Create a set of 10 random booleans
109 random_boolean_set = set(random.choice([True, False]) for _ in range(10))
110
111 # Print the set
112 print(random_boolean_set)
113
114 # Create a tuple of 10 random booleans
115 random_boolean_tuple = tuple(random.choice([True, False]) for _ in range(10))
116
117 # Print the tuple
118 print(random_boolean_tuple)
119
120 # Create a list of 10 random complex numbers
121 random_complexes = [(random.uniform(-10, 10) + random.uniform(-10, 10) * 1j) for _ in range(10)]
122
123 # Print the list
124 print(random_complexes)
125
126 # Create a dictionary with 5 random key-value pairs
127 # The keys are strings and the values are complex numbers
128 random_complex_dict = {}
129 for _ in range(5):
130     key = 'key_' + str(random.randint(0, 999))
131     value = (random.uniform(-10, 10) + random.uniform(-10, 10) * 1j)
132     random_complex_dict[key] = value
133
134 # Print the dictionary
135 print(random_complex_dict)
136
137 # Create a set of 10 random complex numbers
138 random_complex_set = set((random.uniform(-10, 10) + random.uniform(-10, 10) * 1j) for _ in range(10))
139
140 # Print the set
141 print(random_complex_set)
142
143 # Create a tuple of 10 random complex numbers
144 random_complex_tuple = tuple((random.uniform(-10, 10) + random.uniform(-10, 10) * 1j) for _ in range(10))
145
146 # Print the tuple
147 print(random_complex_tuple)
148
149 # Create a list of 10 random lists
150 random_lists = [[random.randint(1, 100) for _ in range(5)] for _ in range(10)]
151
152 # Print the list
153 print(random_lists)
154
155 # Create a dictionary with 5 random key-value pairs
156 # The keys are strings and the values are lists
157 random_list_dict = {}
158 for _ in range(5):
159     key = 'key_' + str(random.randint(0, 999))
160     value = [random.randint(1, 100) for _ in range(5)]
161     random_list_dict[key] = value
162
163 # Print the dictionary
164 print(random_list_dict)
165
166 # Create a set of 10 random lists
167 random_list_set = set([random.randint(1, 100) for _ in range(5)] for _ in range(10))
168
169 # Print the set
170 print(random_list_set)
171
172 # Create a tuple of 10 random lists
173 random_list_tuple = tuple([random.randint(1, 100) for _ in range(5)] for _ in range(10))
174
175 # Print the tuple
176 print(random_list_tuple)
177
178 # Create a list of 10 random dictionaries
179 random_dicts = [{key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10)]
180
181 # Print the list
182 print(random_dicts)
183
184 # Create a dictionary with 5 random key-value pairs
185 # The keys are strings and the values are dictionaries
186 random_dict_dict = {}
187 for _ in range(5):
188     key = 'key_' + str(random.randint(0, 999))
189     value = {key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']}
190     random_dict_dict[key] = value
191
192 # Print the dictionary
193 print(random_dict_dict)
194
195 # Create a set of 10 random dictionaries
196 random_dict_set = set({key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10))
197
198 # Print the set
199 print(random_dict_set)
200
201 # Create a tuple of 10 random dictionaries
202 random_dict_tuple = tuple({key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10))
203
204 # Print the tuple
205 print(random_dict_tuple)
206
207 # Create a list of 10 random sets
208 random_sets = [set(random.randint(1, 100) for _ in range(5)) for _ in range(10)]
209
210 # Print the list
211 print(random_sets)
212
213 # Create a dictionary with 5 random key-value pairs
214 # The keys are strings and the values are sets
215 random_set_dict = {}
216 for _ in range(5):
217     key = 'key_' + str(random.randint(0, 999))
218     value = set(random.randint(1, 100) for _ in range(5))
219     random_set_dict[key] = value
220
221 # Print the dictionary
222 print(random_set_dict)
223
224 # Create a set of 10 random sets
225 random_set_set = set(set(random.randint(1, 100) for _ in range(5)) for _ in range(10))
226
227 # Print the set
228 print(random_set_set)
229
230 # Create a tuple of 10 random sets
231 random_set_tuple = tuple(set(random.randint(1, 100) for _ in range(5)) for _ in range(10))
232
233 # Print the tuple
234 print(random_set_tuple)
235
236 # Create a list of 10 random tuples
237 random_tuples = [tuple(random.randint(1, 100) for _ in range(5)) for _ in range(10)]
238
239 # Print the list
240 print(random_tuples)
241
242 # Create a dictionary with 5 random key-value pairs
243 # The keys are strings and the values are tuples
244 random_tuple_dict = {}
245 for _ in range(5):
246     key = 'key_' + str(random.randint(0, 999))
247     value = tuple(random.randint(1, 100) for _ in range(5))
248     random_tuple_dict[key] = value
249
250 # Print the dictionary
251 print(random_tuple_dict)
252
253 # Create a set of 10 random tuples
254 random_tuple_set = set(tuple(random.randint(1, 100) for _ in range(5)) for _ in range(10))
255
256 # Print the set
257 print(random_tuple_set)
258
259 # Create a tuple of 10 random tuples
260 random_tuple_tuple = tuple(tuple(random.randint(1, 100) for _ in range(5)) for _ in range(10))
261
262 # Print the tuple
263 print(random_tuple_tuple)
264
265 # Create a list of 10 random dictionaries
266 random_dicts = [{key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10)]
267
268 # Print the list
269 print(random_dicts)
270
271 # Create a dictionary with 5 random key-value pairs
272 # The keys are strings and the values are dictionaries
273 random_dict_dict = {}
274 for _ in range(5):
275     key = 'key_' + str(random.randint(0, 999))
276     value = {key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']}
277     random_dict_dict[key] = value
278
279 # Print the dictionary
280 print(random_dict_dict)
281
282 # Create a set of 10 random dictionaries
283 random_dict_set = set({key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10))
284
285 # Print the set
286 print(random_dict_set)
287
288 # Create a tuple of 10 random dictionaries
289 random_dict_tuple = tuple({key: random.randint(1, 100) for key in ['key_1', 'key_2', 'key_3']} for _ in range(10))
290
291 # Print the tuple
292 print(random_dict_tuple)
293
294 # Create a list of 10 random sets
295 random_sets = [set(random.randint(1, 100) for _ in range(5)) for _ in range(10)]
296
297 # Print the list
298 print(random_sets)
299
300 # Create a dictionary with 5 random key-value pairs
301 # The keys are strings and the values are sets
302 random_set_dict = {}
303 for _ in range(5):
304     key = 'key_' + str(random.randint(0, 999))
305     value = set(random.randint(1, 100) for _ in range(5))
306     random_set_dict[key] = value
307
308 # Print the dictionary
309 print(random_set_dict)
310
311 # Create a set of 10 random sets
312 random_set_set = set(set(random.randint(1, 100) for _ in range(5)) for _ in range(10))
313
314 # Print the set
315 print(random_set_set)
316
317 # Create a tuple of 10 random sets
318 random_set_tuple = tuple(set(random.randint(1, 100) for _ in range(5)) for _ in range(10))
319
320 # Print the tuple
321 print(random_set_tuple)
322
323 # Create a list of 10 random tuples
324 random_tuples = [tuple(random.randint(1, 100) for _ in range(5)) for _ in range(10)]
325
326 # Print the list
327 print(random_tuples)
328
329 # Create a dictionary with 5 random key-value pairs
330 # The keys are strings and the values are tuples
331 random_tuple_dict = {}
332 for _ in range(5):
333     key = 'key_' + str(random.randint(0, 999))
334     value = tuple(random.randint(1, 100) for _ in range(5))
335     random_tuple_dict[key] = value
336
337 # Print the dictionary
338 print(random_tuple
```

## StandardSessionManager

```

1  # Import the required modules
2  import pandas as pd
3  import numpy as np
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import mean_squared_error, r2_score
7  from sklearn.linear_model import LinearRegression
8  from sklearn.ensemble import RandomForestRegressor
9  from sklearn.svm import SVR
10 from sklearn.neighbors import KNeighborsRegressor
11
12 # Load the dataset
13 data = pd.read_csv('data.csv')
14
15 # Split the data into features and target variable
16 X = data[['feature1', 'feature2', 'feature3']]
17 y = data['target']
18
19 # Split the data into training and testing sets
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # Standardize the features
23 scaler = StandardScaler()
24 X_train = scaler.fit_transform(X_train)
25 X_test = scaler.transform(X_test)
26
27 # Train the Linear Regression model
28 lr = LinearRegression()
29 lr.fit(X_train, y_train)
30
31 # Predict using the Linear Regression model
32 y_pred_lr = lr.predict(X_test)
33
34 # Calculate the Mean Squared Error (MSE) and R-squared (R2) for Linear Regression
35 mse_lr = mean_squared_error(y_test, y_pred_lr)
36 r2_lr = r2_score(y_test, y_pred_lr)
37
38 # Train the Random Forest model
39 rf = RandomForestRegressor()
40 rf.fit(X_train, y_train)
41
42 # Predict using the Random Forest model
43 y_pred_rf = rf.predict(X_test)
44
45 # Calculate the Mean Squared Error (MSE) and R-squared (R2) for Random Forest
46 mse_rf = mean_squared_error(y_test, y_pred_rf)
47 r2_rf = r2_score(y_test, y_pred_rf)
48
49 # Train the Support Vector Regression (SVR) model
50 svr = SVR()
51 svr.fit(X_train, y_train)
52
53 # Predict using the SVR model
54 y_pred_svr = svr.predict(X_test)
55
56 # Calculate the Mean Squared Error (MSE) and R-squared (R2) for SVR
57 mse_svr = mean_squared_error(y_test, y_pred_svr)
58 r2_svr = r2_score(y_test, y_pred_svr)
59
60 # Train the K-Nearest Neighbors (KNN) model
61 knn = KNeighborsRegressor()
62 knn.fit(X_train, y_train)
63
64 # Predict using the KNN model
65 y_pred_knn = knn.predict(X_test)
66
67 # Calculate the Mean Squared Error (MSE) and R-squared (R2) for KNN
68 mse_knn = mean_squared_error(y_test, y_pred_knn)
69 r2_knn = r2_score(y_test, y_pred_knn)
70
71 # Print the results
72 print("Linear Regression Results:")
73 print("MSE: ", mse_lr)
74 print("R2: ", r2_lr)
75
76 print("Random Forest Results:")
77 print("MSE: ", mse_rf)
78 print("R2: ", r2_rf)
79
80 print("SVR Results:")
81 print("MSE: ", mse_svr)
82 print("R2: ", r2_svr)
83
84 print("KNN Results:")
85 print("MSE: ", mse_knn)
86 print("R2: ", r2_knn)
87
88 # End of the program

```

## ServerSession

```

1  # Import the required modules
2  import pandas as pd
3  import numpy as np
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import mean_squared_error, r2_score
7  from sklearn.linear_model import LinearRegression
8  from sklearn.ensemble import RandomForestRegressor
9  from sklearn.svm import SVR
10 from sklearn.neural_network import MLPRegressor
11
12 # Load the dataset
13 data = pd.read_csv('data.csv')
14
15 # Split the data into features and target variable
16 X = data[['feature1', 'feature2', 'feature3']]
17 y = data['target']
18
19 # Split the data into training and testing sets
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # Standardize the features
23 scaler = StandardScaler()
24 X_train = scaler.fit_transform(X_train)
25 X_test = scaler.transform(X_test)
26
27 # Train the Linear Regression model
28 lr = LinearRegression()
29 lr.fit(X_train, y_train)
30
31 # Predict using Linear Regression
32 y_lr_pred = lr.predict(X_test)
33
34 # Train the Random Forest model
35 rf = RandomForestRegressor()
36 rf.fit(X_train, y_train)
37
38 # Predict using Random Forest
39 y_rf_pred = rf.predict(X_test)
40
41 # Train the SVM model
42 svm = SVR()
43 svm.fit(X_train, y_train)
44
45 # Predict using SVM
46 y_svm_pred = svm.predict(X_test)
47
48 # Train the MLP model
49 mlp = MLPRegressor()
50 mlp.fit(X_train, y_train)
51
52 # Predict using MLP
53 y_mlp_pred = mlp.predict(X_test)
54
55 # Evaluate the models
56 lr_rmse = mean_squared_error(y_test, y_lr_pred)
57 rf_rmse = mean_squared_error(y_test, y_rf_pred)
58 svm_rmse = mean_squared_error(y_test, y_svm_pred)
59 mlp_rmse = mean_squared_error(y_test, y_mlp_pred)
60
61 lr_r2 = r2_score(y_test, y_lr_pred)
62 rf_r2 = r2_score(y_test, y_rf_pred)
63 svm_r2 = r2_score(y_test, y_svm_pred)
64 mlp_r2 = r2_score(y_test, y_mlp_pred)
65
66 # Print the results
67 print("Linear Regression RMSE: ", lr_rmse)
68 print("Random Forest RMSE: ", rf_rmse)
69 print("SVM RMSE: ", svm_rmse)
70 print("MLP RMSE: ", mlp_rmse)
71
72 print("Linear Regression R2: ", lr_r2)
73 print("Random Forest R2: ", rf_r2)
74 print("SVM R2: ", svm_r2)
75 print("MLP R2: ", mlp_r2)
76
77 # End of the program

```

## ServerSessionManager

[illegible]

aspectj.org

# Stan Wagon's bike

*My square-wheel bike, on permanent display at Macalester College. This construction, believe it or not, earned me an entry in "Ripley's Believe It or Not"; beats standing in a block of ice for three days or growing three-foot long fingernails.*

--

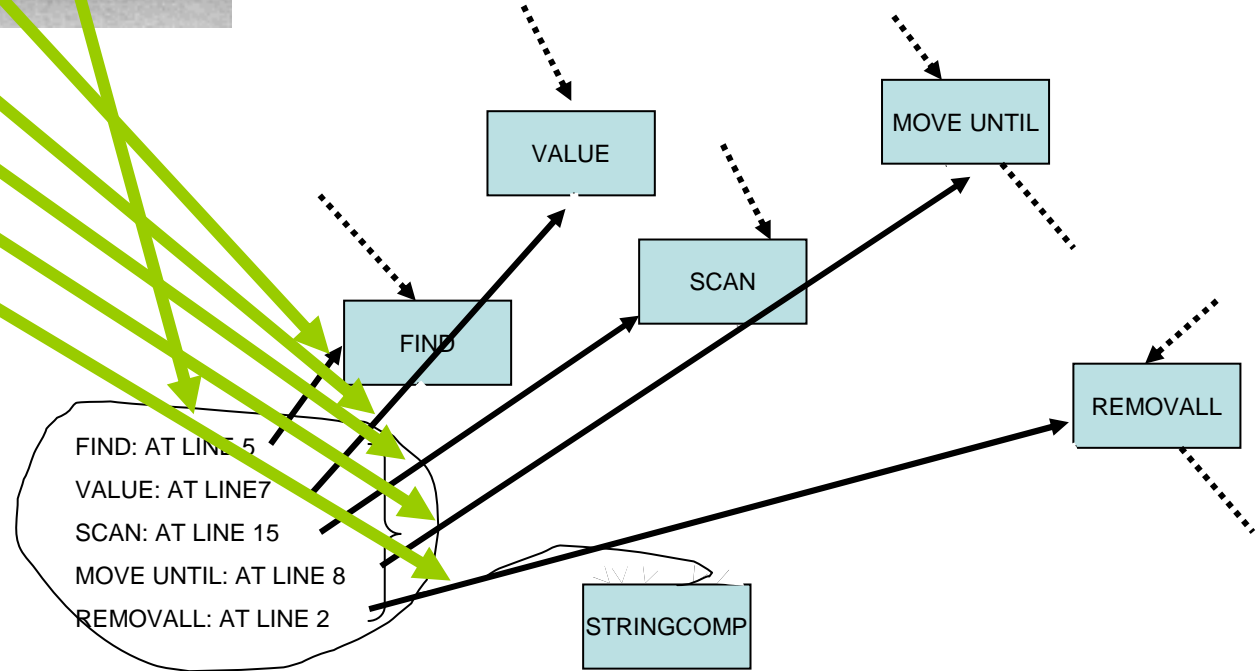
<http://www.stanwagon.com>

**Stan Wagon**  
[\(\[wagon@macalester.edu\]\(mailto:wagon@macalester.edu\)\)](mailto:wagon@macalester.edu), Prof. of  
Mathematics and  
Computer Science,  
Macalester College,  
St. Paul, Minnesota

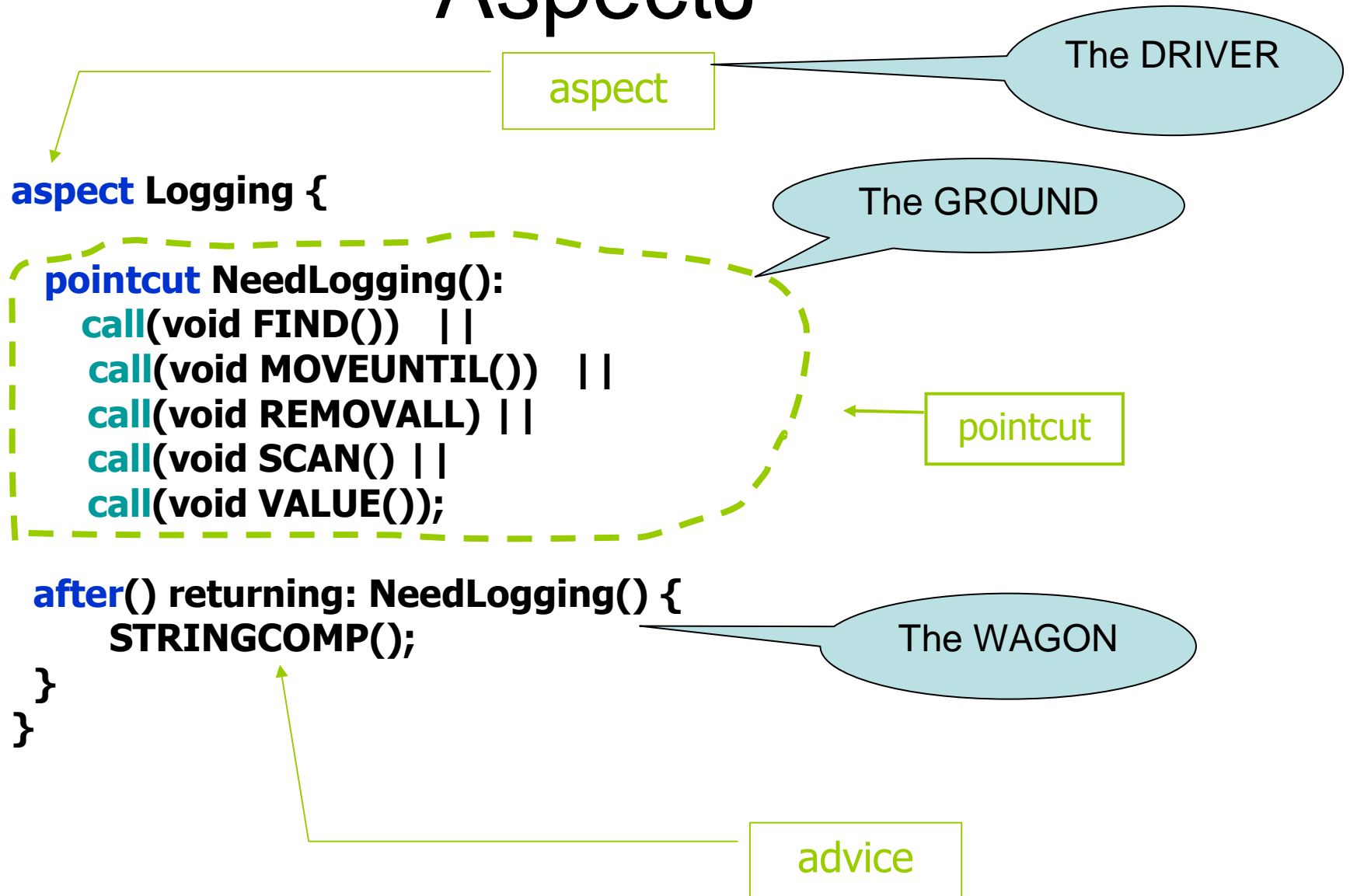




## *The Weaver*



# AspectJ





## 2. Aspect-orientation at large

# 2.1 Aspect-oriented Programming

- It permeates into almost every popular high-level programming languages

- Java

[Hyper/J](#), [AspectJ](#), [AJDT](#), [JBoss](#)

- C/C++/C#

[AspectC/C++](#), [C#](#)

- PHP

[AOPHP](#), [AspectPHP](#)

... and many many more: see [AOSD.NET](#)

# Every AOP mechanism has to support

- Definition and representation of aspects
  - Definition of Advices in the component language
  - Definition of Joinpoints in regular expressions
    - *Optionally, they can introduce new data members, changing the structures of components*
  - *Representation: New keywords, New directives, XML, but never change the code of components directly*
- Implementing a weaver
  - As preprocessor => generates woven components in the component language (AspectC, AOPHP)
  - As instrumenting compiler => generates woven components in the bytecode for the languages supporting reflection (AspectJ)
  - As interpretator => interpreting the woven code on-the-fly (AspectPHP)

## 2. Aspect-orientation at large

# 2.2 Aspect-Oriented SD

- AO includes the whole lifecycle of SE
  - <http://www.aosd.net>
- There is a conference AOSD
- There are workshops on Early Aspects at AOSD, OOPSLA, ICSE
- Hot topics related to all other SD technologies
  - Aspect-oriented Refactoring
  - Aspect Mining
  - Aspect-oriented Debugging
  - Aspect-oriented Testing
  - Aspect-oriented Slicing
  - Aspect-oriented Model Checking
- ...

## 2. Aspect-orientation at large

# 2.3 Aspect-Oriented RE

- Lessons learnt from success stories
  - SP  $\Rightarrow$  SA
  - OOP  $\Rightarrow$  OOA
  - Why not AOP  $\Rightarrow$  AOA?
    - Separation of crosscutting concerns earlier
    - Avoid duplication as early as possible
    - Identify aspects before mining them from code
- Discover aspects in the early requirements
  - From structured requirement documents
  - From unstructured (textual) documents
- Verify discovered (candidate) aspects in AOP

### 3. A Case Study on AORE

1. Quickly go through goal-oriented requirements engineering basics
2. A requirements engineering process to elicit early aspects (goal aspects)
3. A reverse engineering exercise to identify candidate aspects (code aspects)
4. Linking goal aspects with code aspects

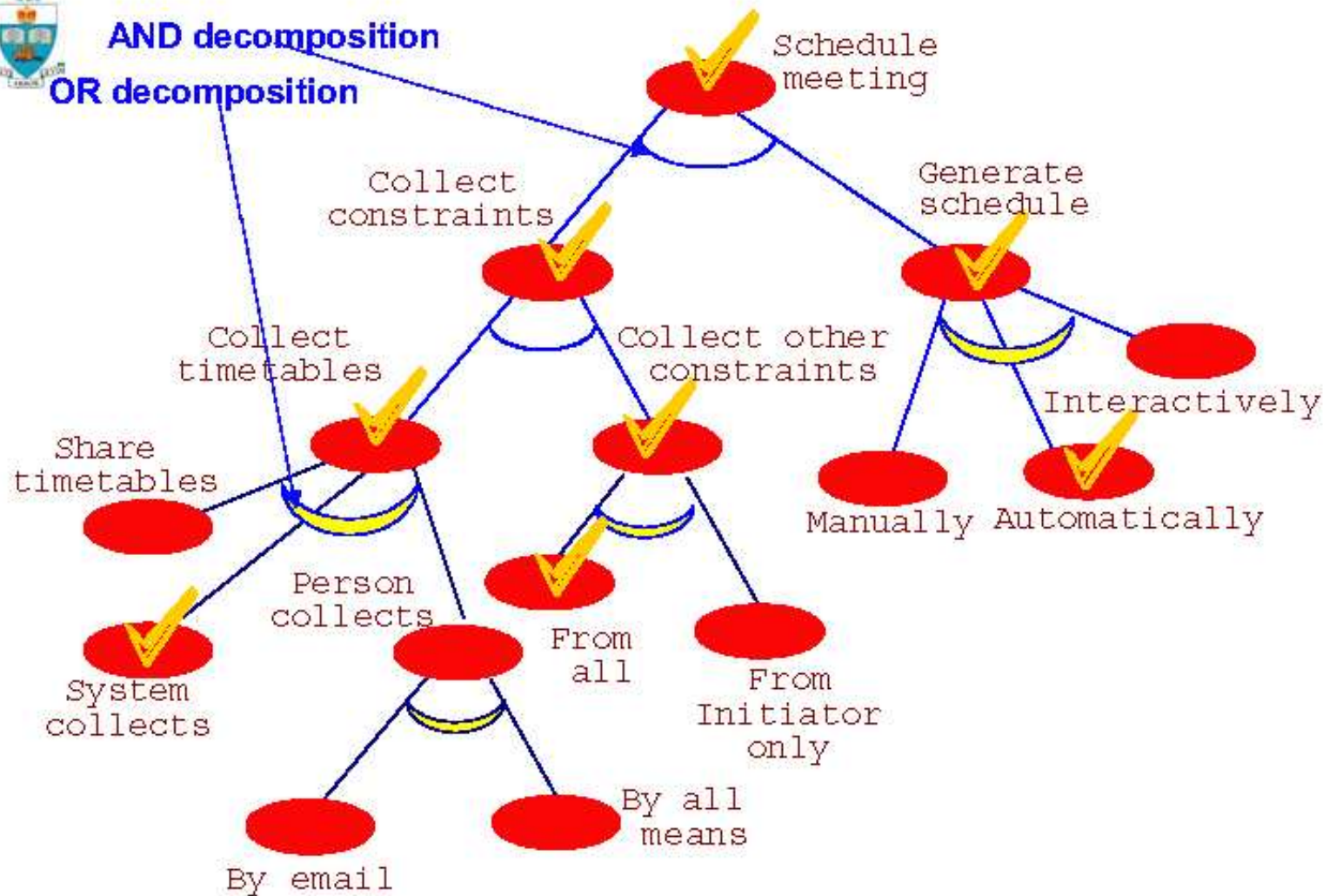
# 3.1 Requirements Goal Models

- A goal model is an intentional model
- A goal can be decomposed into AND or OR subgoals
- A goal model has both hard and soft goals
  - A hard goal can be either satisfied or denied
  - A soft goal is partially satisfied => *satisficed*
- Soft goal uses HELP (+), HURT (-), MAKE (++) or BREAK (--) correlations to show partial satisfaction (satisfice) from a set of subgoals

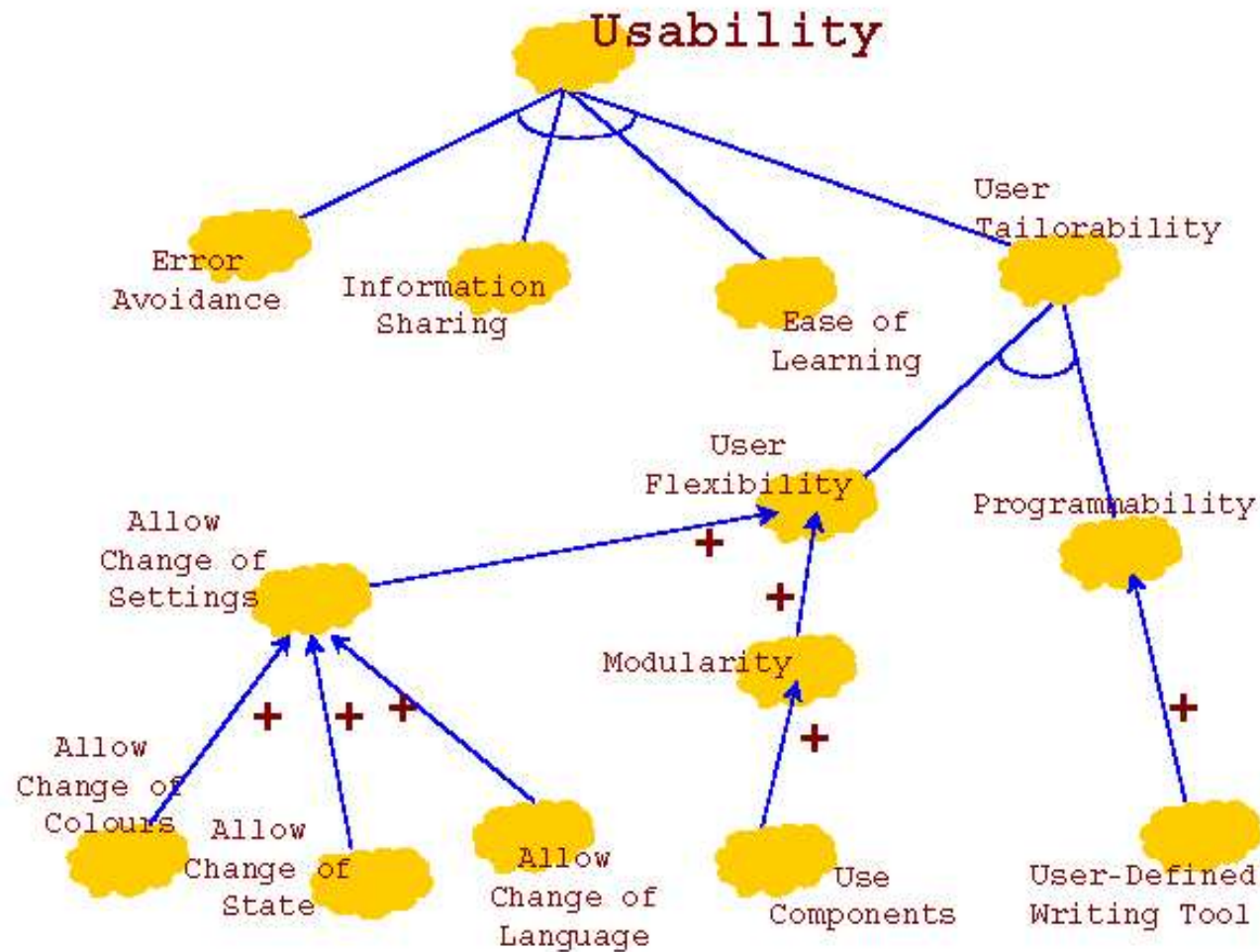
# 3.1.1 Hard goal model



**AND decomposition**  
**OR decomposition**

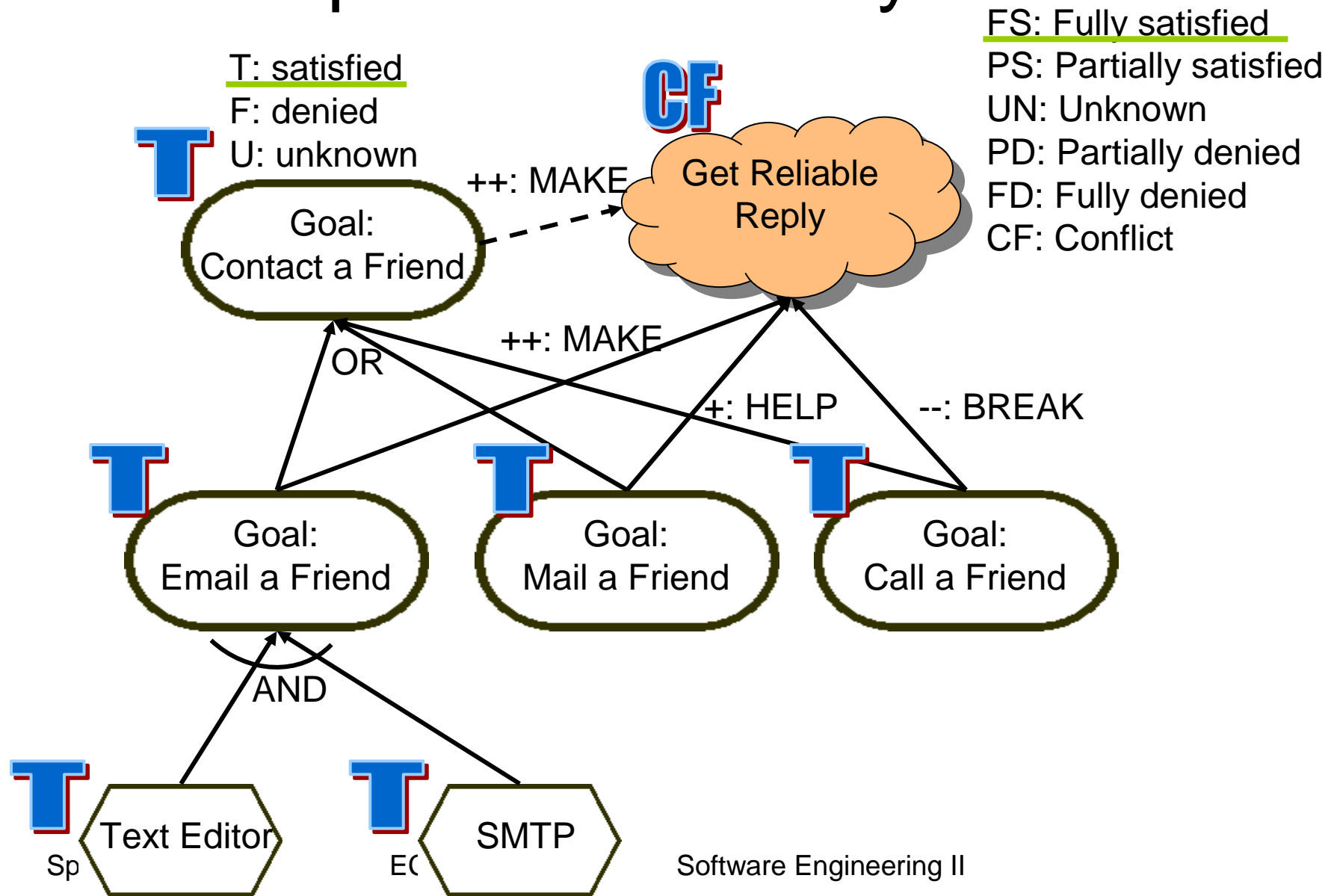


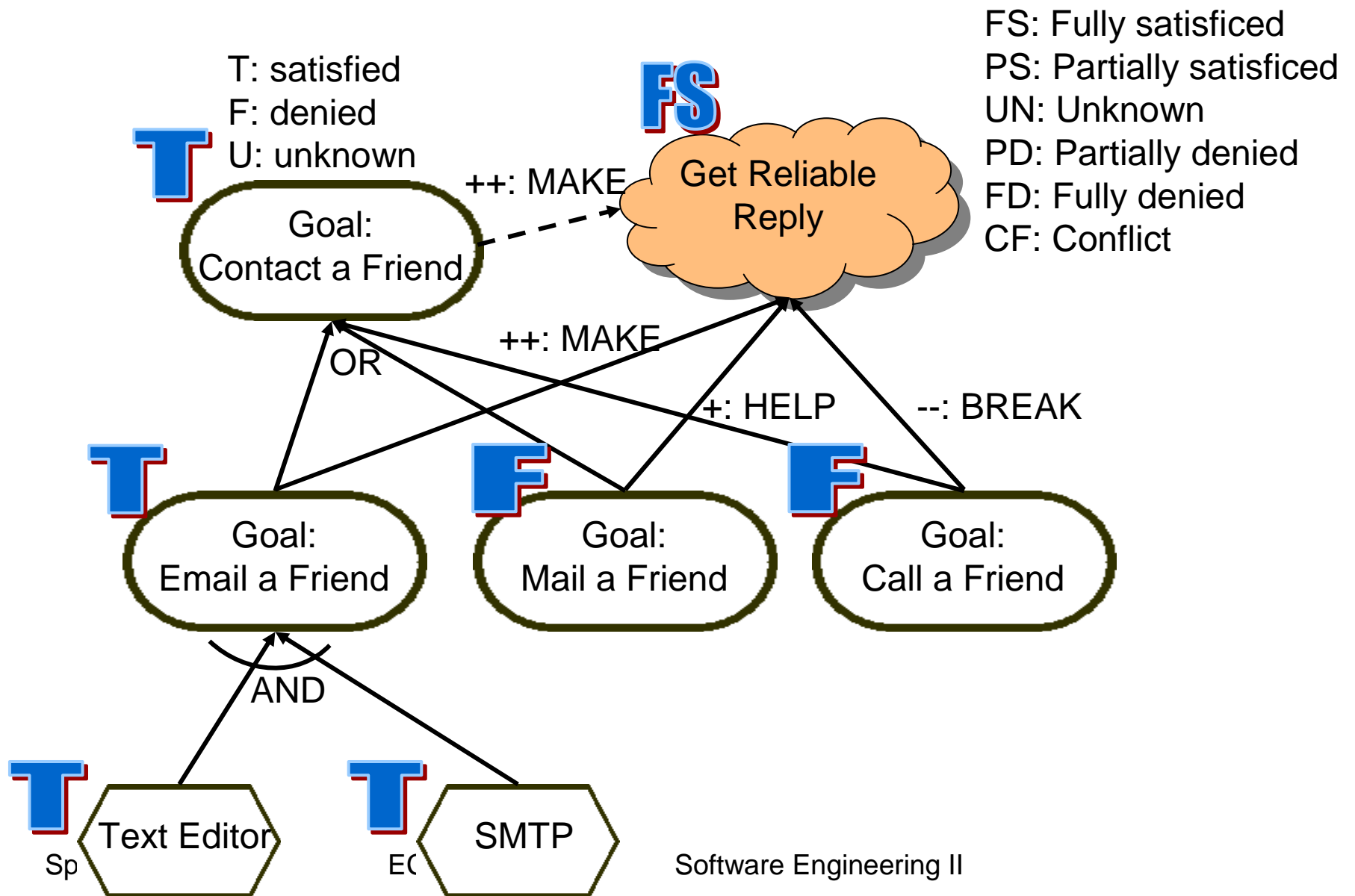
## 3.1.2 Soft goal model





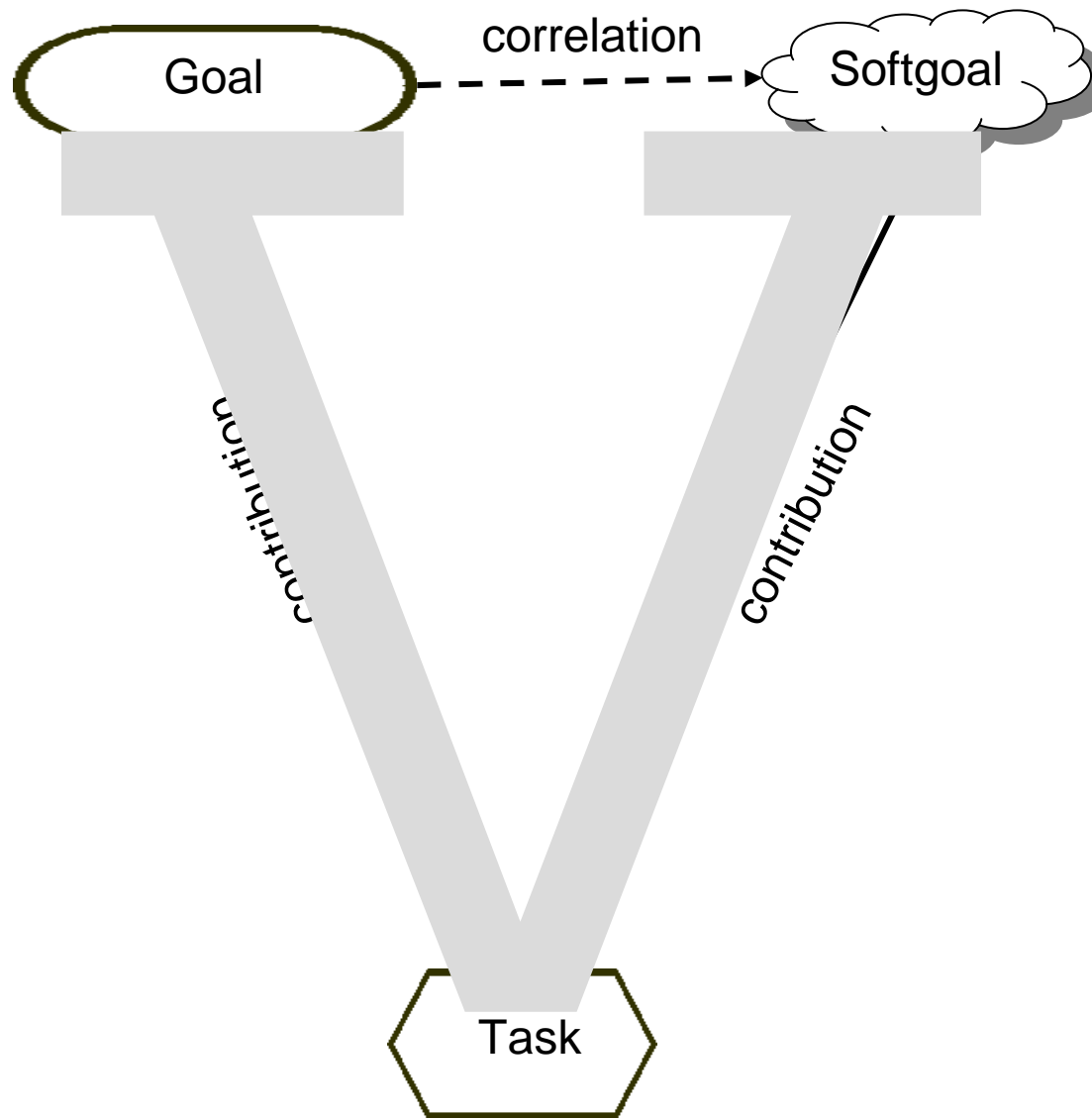
# 3.1.3 Goal-Oriented Requirements Analysis





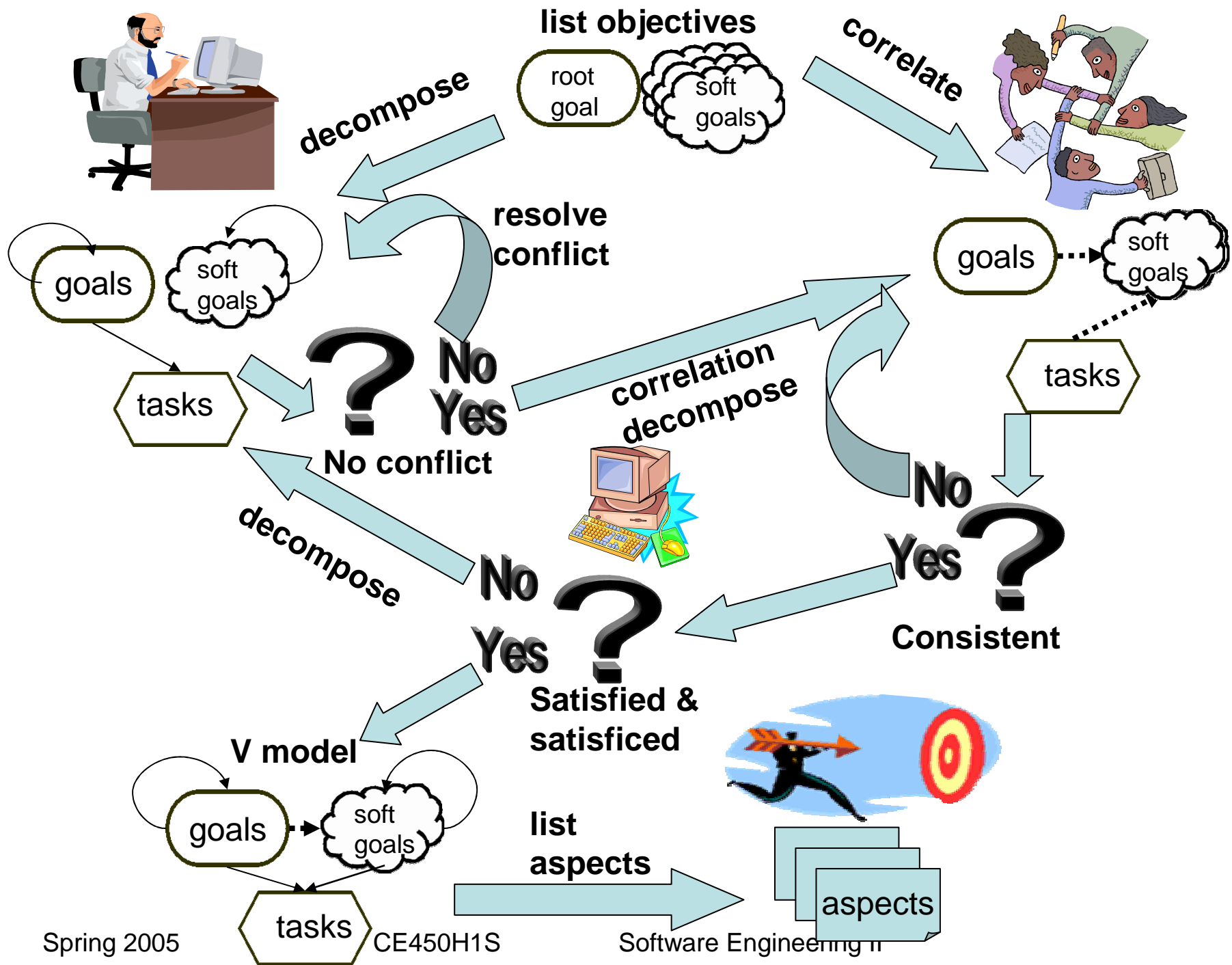
## 3.1.4 V-graph

In order to reason about interplay of functional and non-functional requirements, we create a particular type of goal model, called *V-graph*



## 3.2 The Process

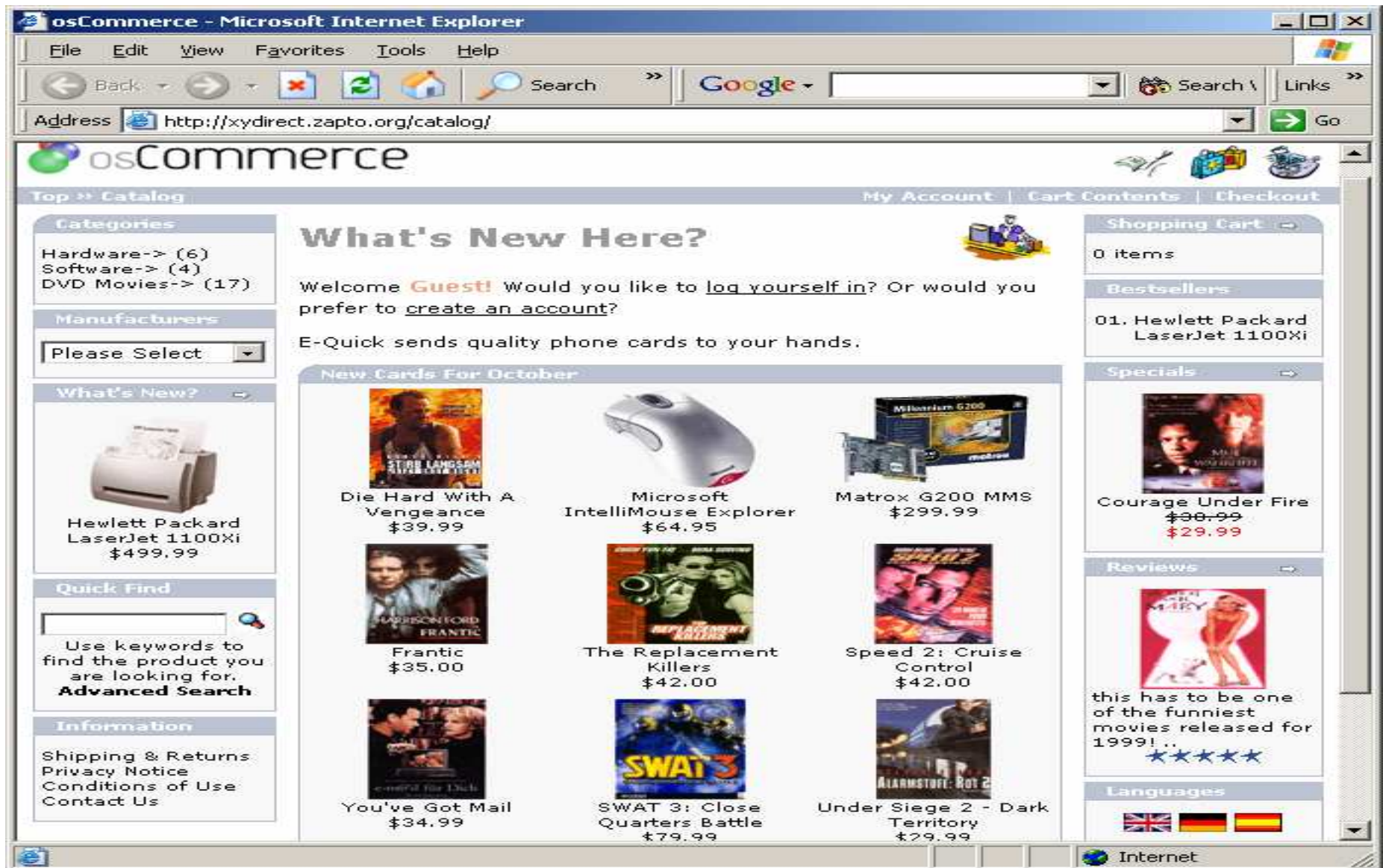
- *Start from root-level goals and soft goals, correlate and decompose them into a V-graph*
- *A goal analysis based on the label propagation algorithm is used to check for:*
  - *Conflicts*
  - *Inconsistencies*
  - *Denial of any goal or soft goals*
- *After resolving the problems, a proper V-graph is obtained*
- *Then we list the candidate aspects from the V-graph*



## 3.3 A Case Study

- Medi@Shop adapted from literature:  
*Castro, Kolp, Mylopoulos, Towards requirements-driven information systems engineering: the Tropos project, Journal of Information Systems, 2002.*  
*Can we find aspects from early requirements?*
- osCommerce studied from an LAMP (Linux, Apache, MySQL, PHP) Open-Source project:  
(<http://www.oscommerce.com>)  
*Do they manifest in the developed software?*

# osCommerce (version 2.2m2)





# Duplications in code

```
1: <?php
2: /*
3:  $Id: privacy.php,v 1.22 2003/06/05 23:26:23 hpd1 Exp $
4:
5:  osCommerce, Open Source E-Commerce Solutions
6:  http://www.oscommerce.com
7:
8:  Copyright (c) 2003 osCommerce
9:
10: Released under the GNU General Public License
11: */
12:
13: require('includes/application_top.php');
14:
15: require(DIR_WS_LANGUAGES . $language . '/' . FILENAME_PRIVACY);
16:
17: $breadcrumb->add(NAVBAR_TITLE, tep_href_link(FILENAME_PRIVACY));
18: ?>
19: <!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN"
20: <html <?php echo HTML_PARAMS; ?>>
21: <head>
22: <meta http-equiv="Content-Type" content="text/html; charset=<?php
23: <title><?php echo TITLE; ?></title>
24: <base href="<?php echo (($request_type == 'SSL') ? HTTPS_SERVER : HT
25: <link rel="stylesheet" type="text/css" href="stylesheet.css">
26: </head>
27: <body marginwidth="0" marginheight="0" topmargin="0" bottommargin="0
28: <!-- header //-->
29: <?php require(DIR_WS_INCLUDES . 'header.php'); ?>
30: <!-- header_eof //-->
31:
32: <!-- body //-->
33: <table border="0" width="100%" cellspacing="3" cellpadding="3">
34:   <tr>
35:     <td width="<?php echo BOX_WIDTH; ?>" valign="top"><table borde
36: <!-- left_navigation //-->
37: <?php require(DIR_WS_INCLUDES . 'column_left.php'); ?>
38: <!-- left_navigation_eof //-->
39:   </table></td>
40: <!-- body_text //-->
41:   <td width="100%" valign="top"><table border="0" width="100%" c
42:     <tr>
43:       <td><table border="0" width="100%" cellspacing="0" cellpad
44:         <tr>
45:           <td class="pageHeading"><?php echo HEADING_TITLE; ?></td>
```

```
1: <?php
2: /*
3:  $Id: shopping_cart.php,v 1.73 2003/06/09 23:03:56 hpd1 Exp $
4:
5:  osCommerce, Open Source E-Commerce Solutions
6:  http://www.oscommerce.com
7:
8:  Copyright (c) 2003 osCommerce
9:
10: Released under the GNU General Public License
11: */
12:
13: require('includes/application_top.php');
14:
15: require(DIR_WS_LANGUAGES . $language . '/' . FILENAME_SHOPPING_CAR
16:
17: $breadcrumb->add(NAVBAR_TITLE, tep_href_link(FILENAME_SHOPPING_CAR
18: ?>
19: <!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN"
20: <html <?php echo HTML_PARAMS; ?>>
21: <head>
22: <meta http-equiv="Content-Type" content="text/html; charset=<?php ec
23: <title><?php echo TITLE; ?></title>
24: <base href="<?php echo (($request_type == 'SSL') ? HTTPS_SERVER : HT
25: <link rel="stylesheet" type="text/css" href="stylesheet.css">
26: </head>
27: <body marginwidth="0" marginheight="0" topmargin="0" bottommargin="0
28: <!-- header //-->
29: <?php require(DIR_WS_INCLUDES . 'header.php'); ?>
30: <!-- header_eof //-->
31:
32: <!-- body //-->
33: <table border="0" width="100%" cellspacing="3" cellpadding="3">
34:   <tr>
35:     <td width="<?php echo BOX_WIDTH; ?>" valign="top"><table border=
36: <!-- left_navigation //-->
37: <?php require(DIR_WS_INCLUDES . 'column_left.php'); ?>
38: <!-- left_navigation_eof //-->
39:   </table></td>
40: <!-- body_text //-->
41:   <td width="100%" valign="top"><?php echo tep_draw_form('cart_qua
42:     <tr>
43:       <td><table border="0" width="100%" cellspacing="0" cellpaddi
44:         <tr>
45:           <td class="pageHeading"><?php echo HEADING_TITLE; ?></td>
```

10 differences found

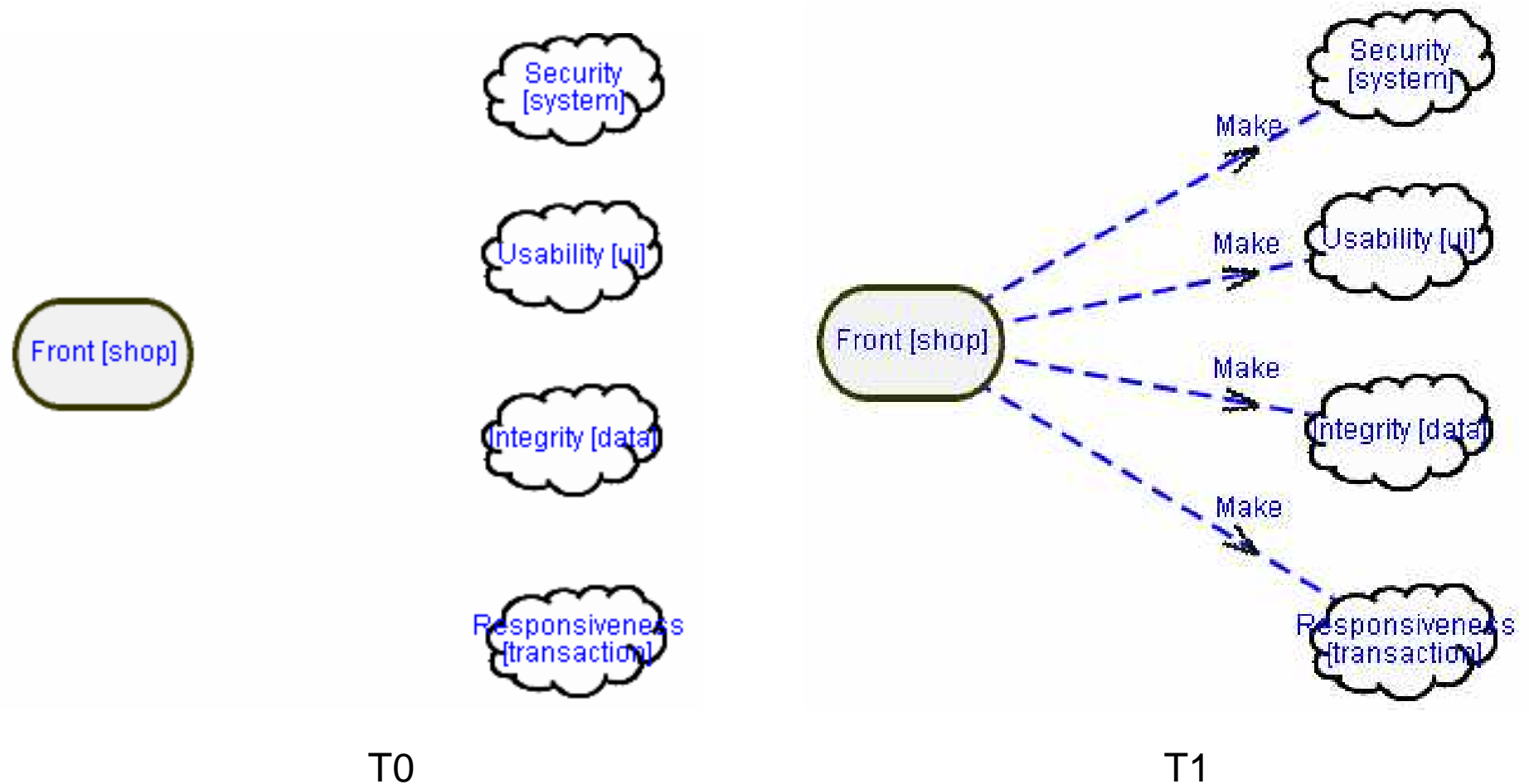
# Candidate code aspects in the code

## Clone detection (by Semantic Design, Inc)

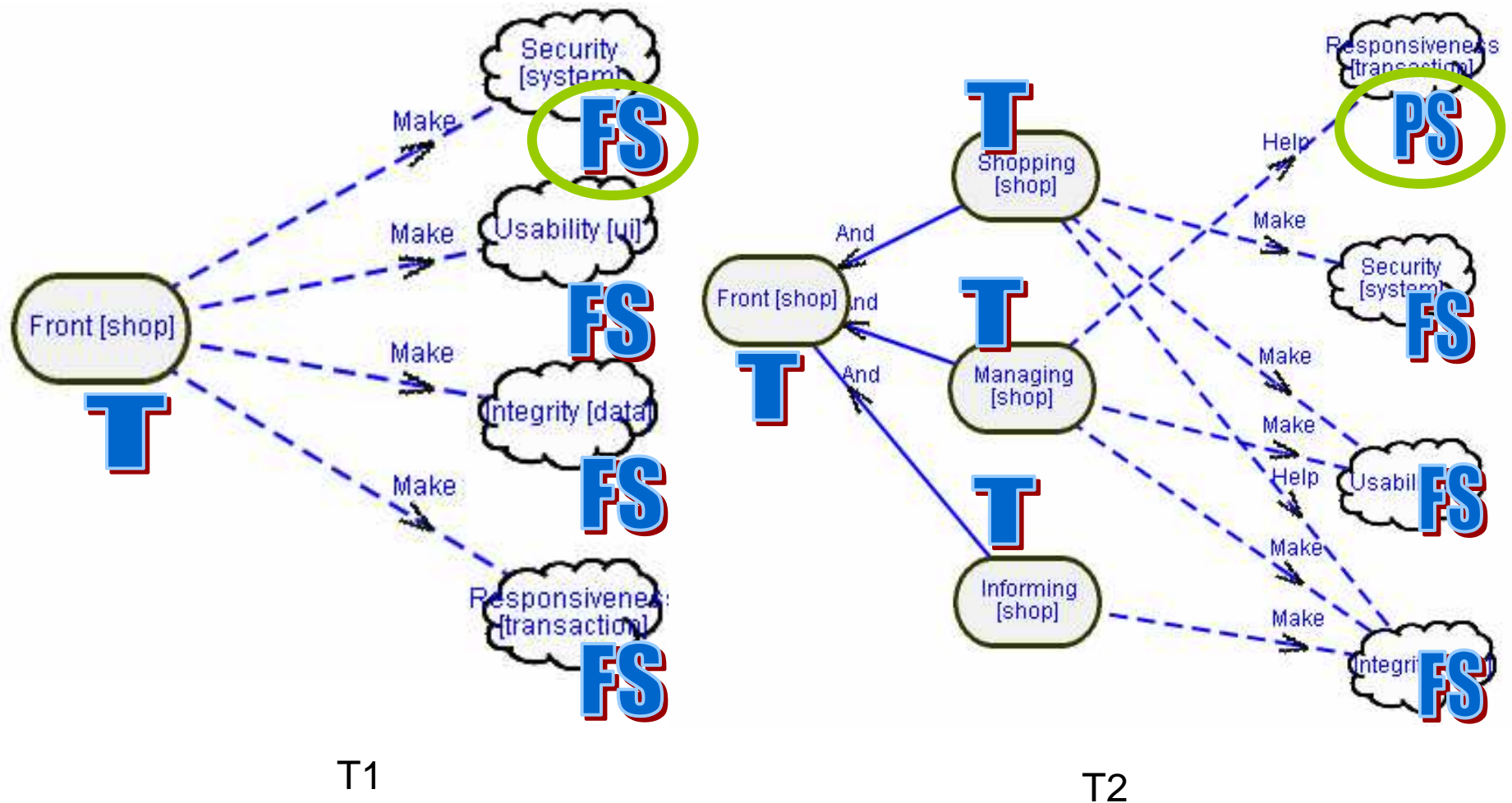
LOC	#clones	Code description	Need refactoring?
1	319	require(\$path . \$file);	No
1	260	echo \$expression;	No
559	2	class email;	No
2	292	define (\$variable, \$value);	No
76	2	class mime;	No
4	67	messageStack->add (\$error);	Yes (NFR)
15	15	Postal code zone check	Yes (FR)
22	10	require(application_top.php); SSL check	Yes (FR/NFR)
3	64	Set HTML head CHARSET	Yes (NFR)

# 3.4 Identifying goal aspects

## Correlate initial goals and softgoals

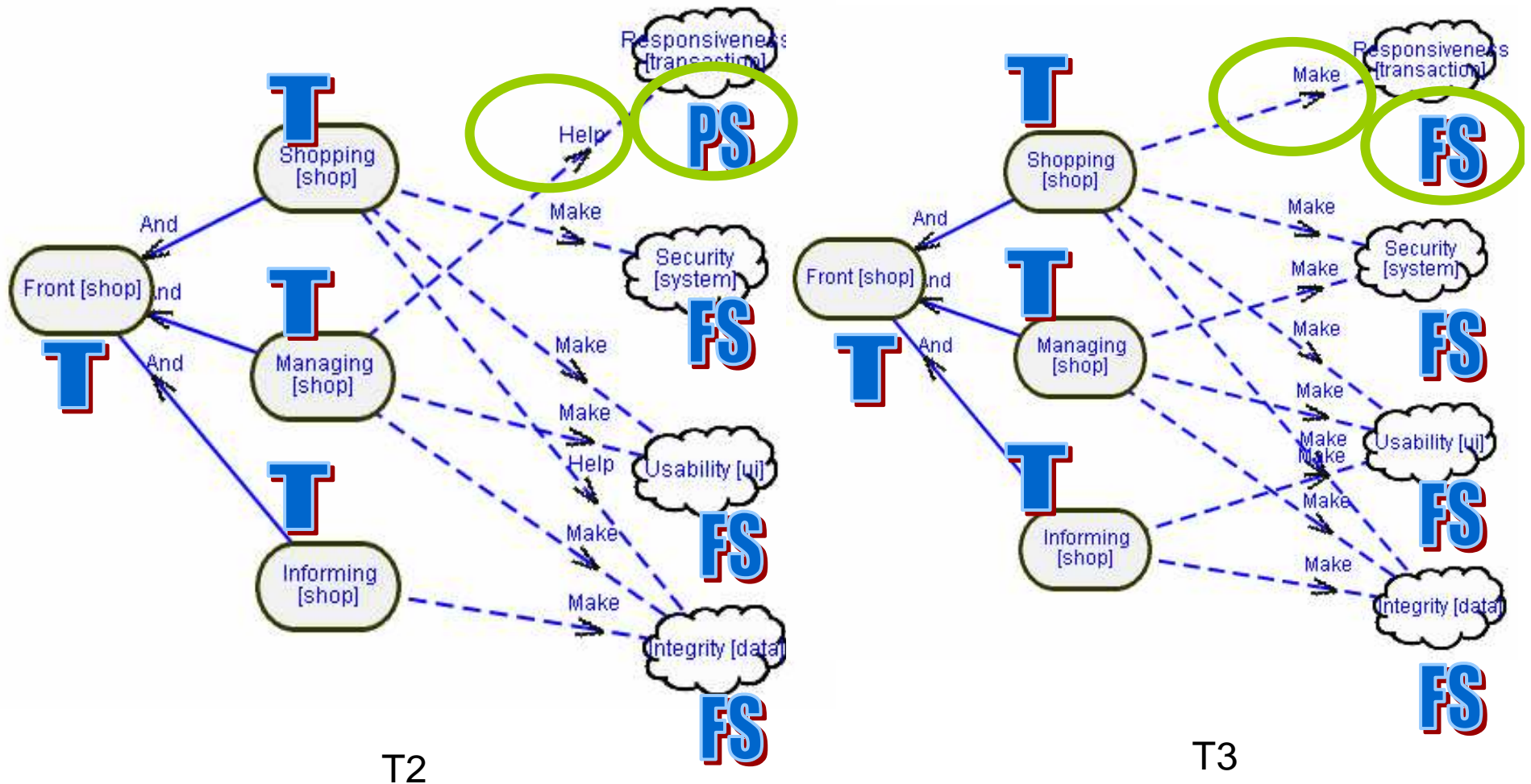


# Inconsistent decomposition

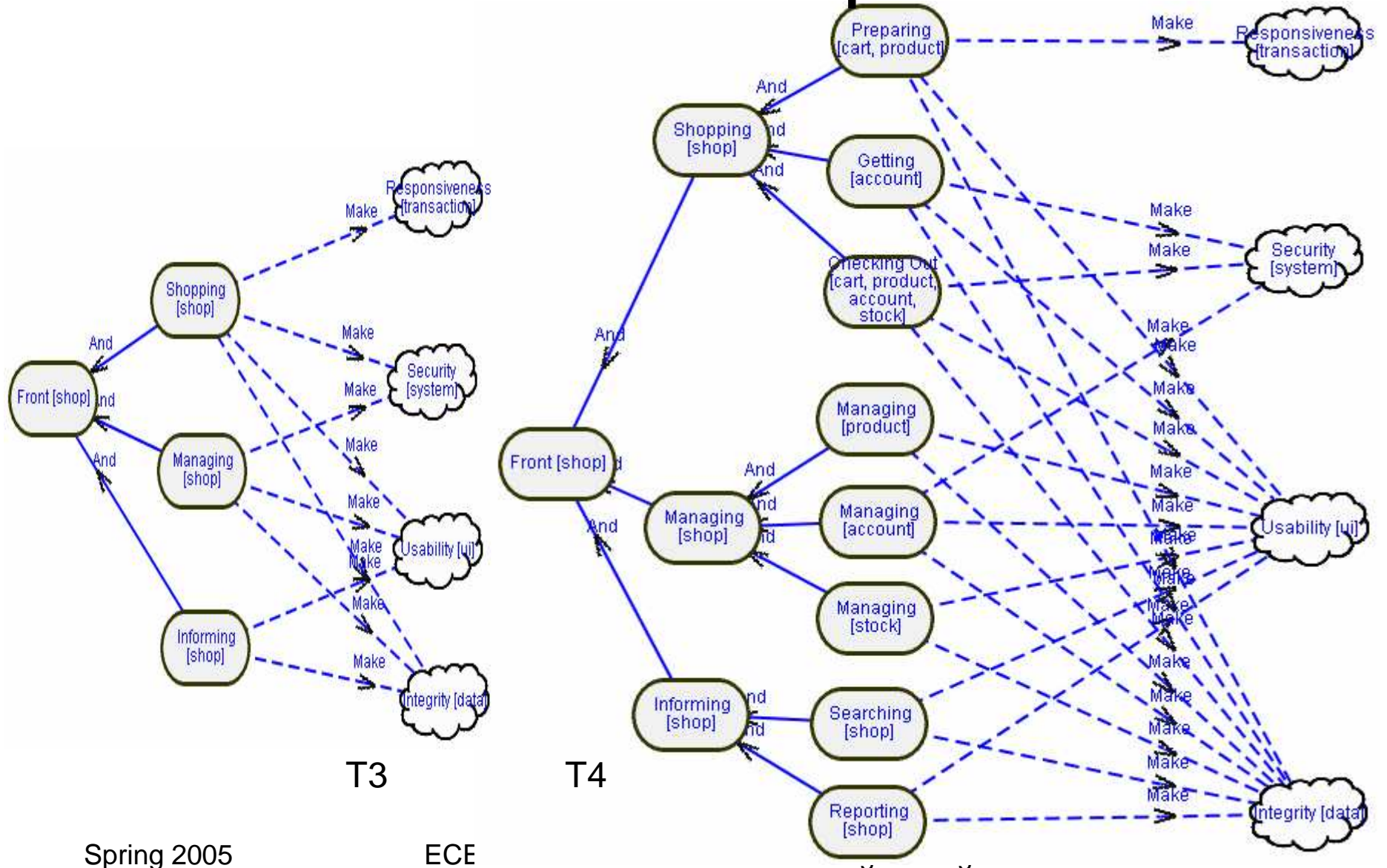




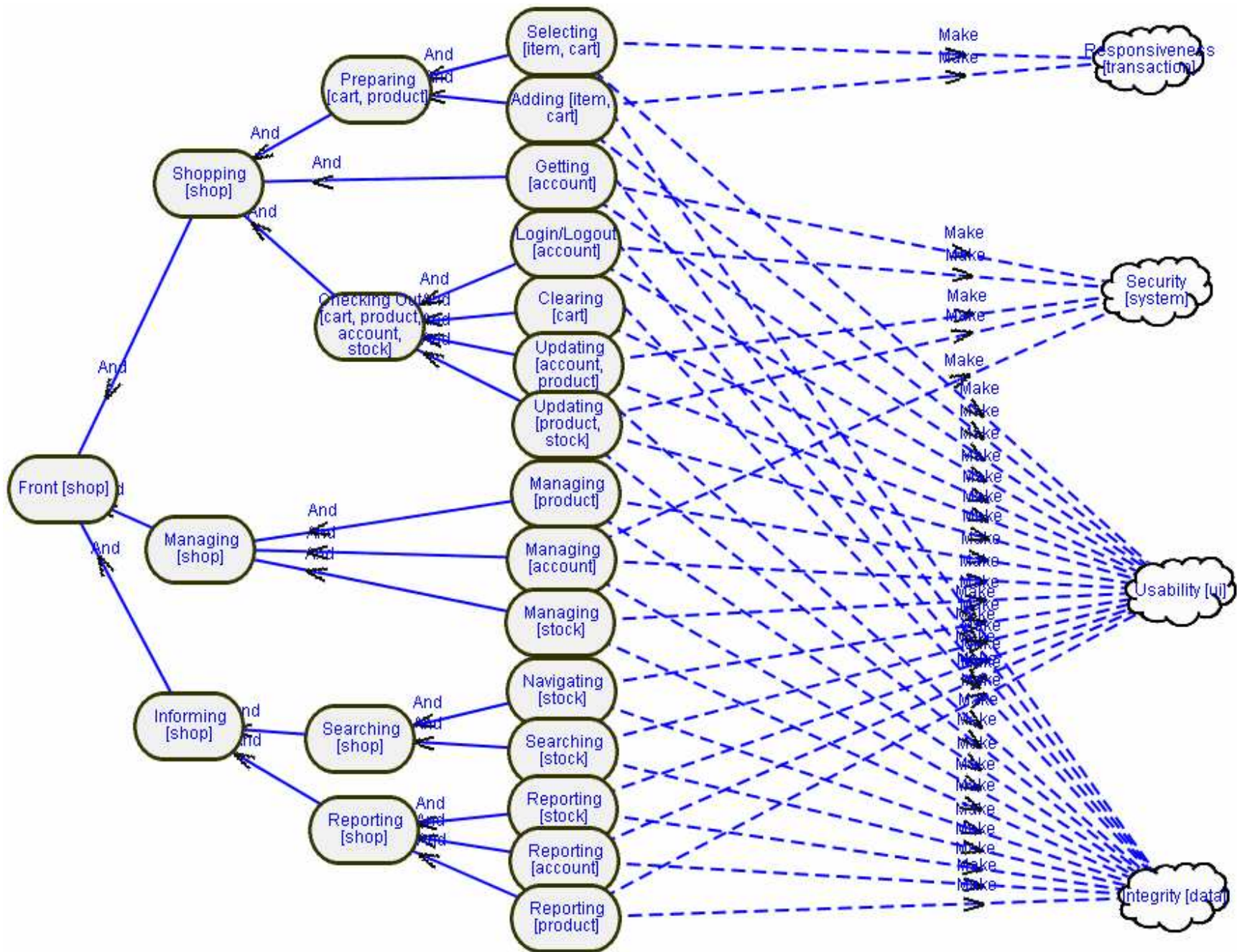
# Resolving inconsistency

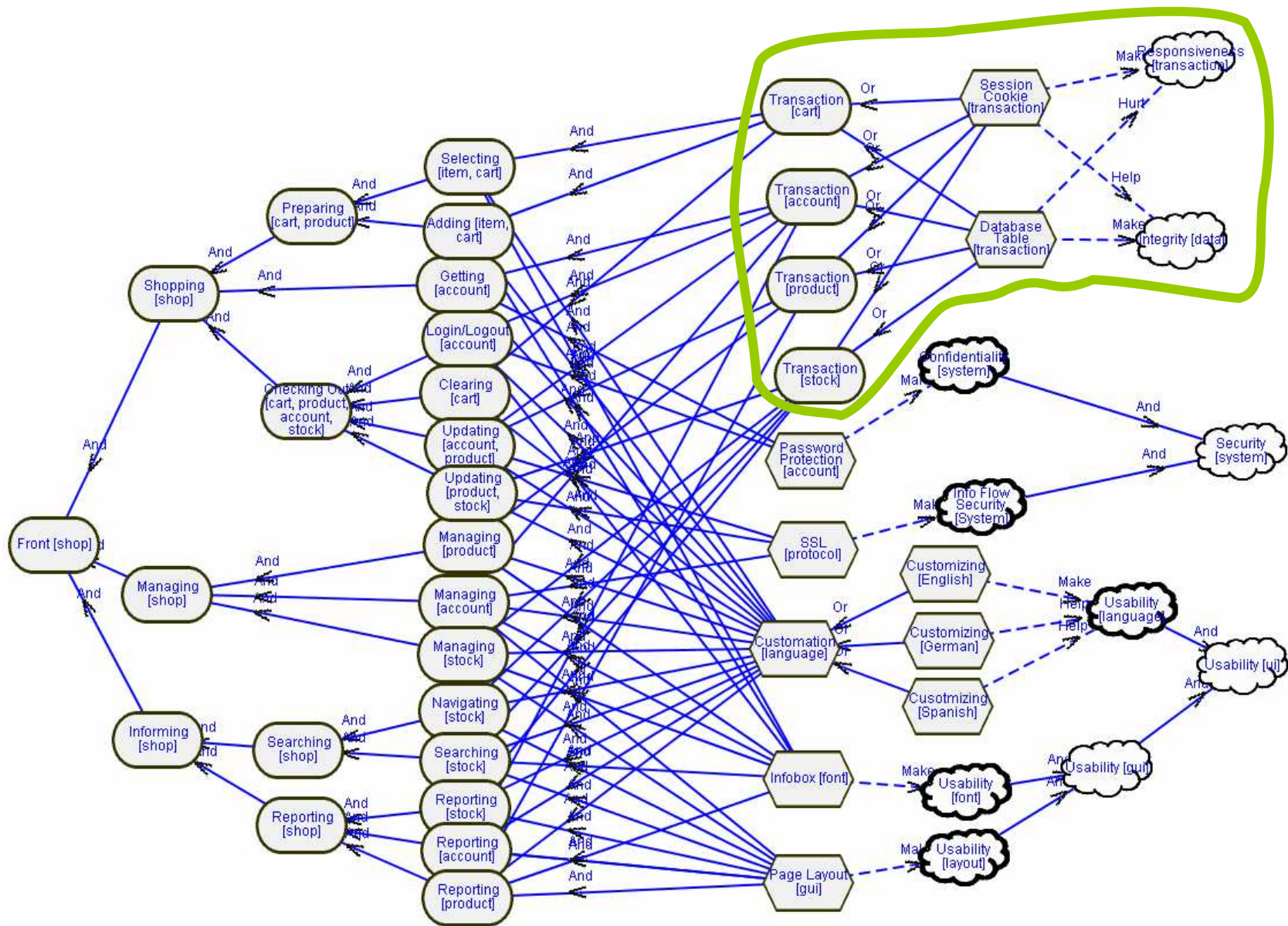


# Further decomposition



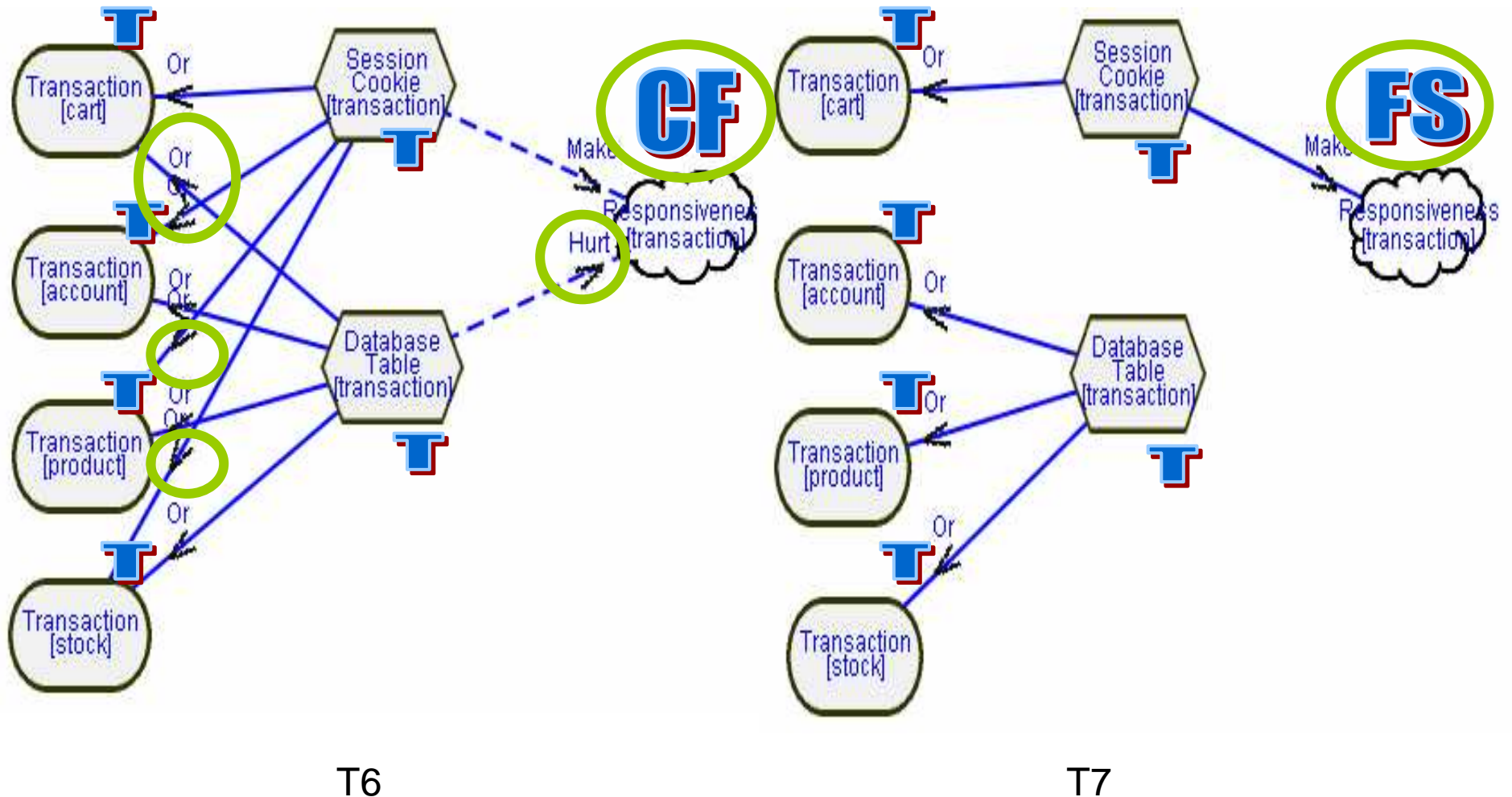








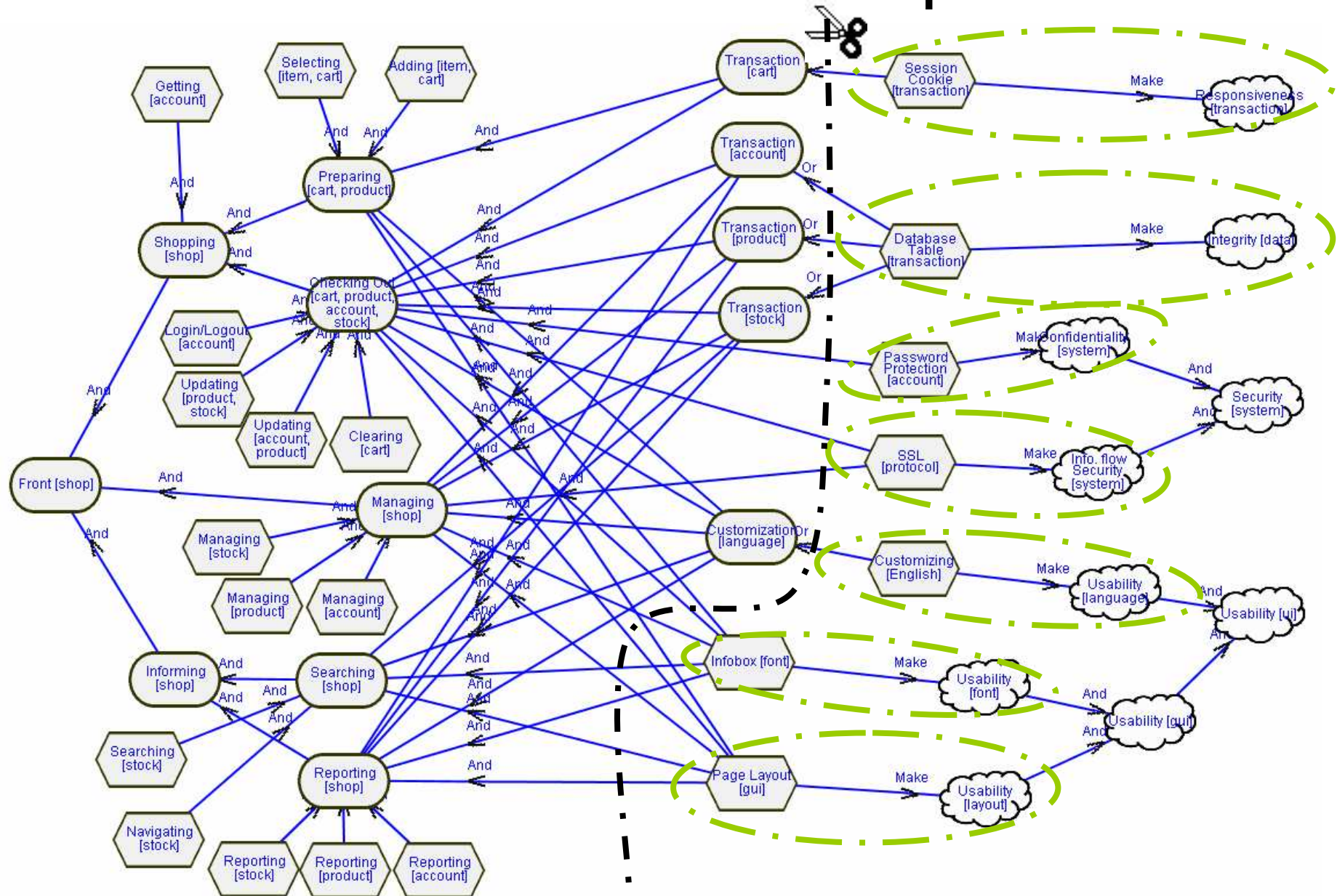
# Resolving Conflicts



T6

T7

# Result candidate aspects



# Goal Aspects

```
goal aspect Responsiveness[transaction] {  
  pointcut transaction():  
    Preparing[cart,product]) ||  
    CheckingOut[cart, product, account, stock]);  
  required () by: transaction() {  
    SessionCookie[transaction]();  
  }  
};
```

- AspectJ-like syntax
- Allow weaving the operationalized tasks with goals specified in the pointcut

# Your exercise

- Reverse Engineering  
Identify some aspects in the OpenOME
  - Clone-detection or Callgraph extraction
  - Goal analysis
- Forward Engineering
  - Implement some new NFR through AspectJ

## 4. Summary

- The concepts of aspect-orientation
- The practise of AOP, AOSD, AORE, AOSR
- A Case study of AORE

# Further readings

- [AOP] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. “Aspect oriented programming”. *LNCS*, 1241:220--242, Oct. 1997.
- [AOPRefactoring] C. Zhang, H.-A. Jacobsen. “Refactoring Middleware with Aspects”. *TPDS* 14(1):1058-1073. 2003
- [AOPMining] C. Zhang, H.-A. Jacobsen. “PRISM is research in Aspect Mining”. *OOPSLA*, 2004.
- [AOPRE] Y. Yu, J.C. Leite, J. Mylopoulos. “From goals to aspects: discovering aspects from goal models”. *RE'04*, 2004.

# What's next ...

- A tutorial on aspect-oriented programming tools
  - AspectJ
  - Eclipse/AJDT
  - Visualizing Aspects
  - Aspect mining tool
- A lecture on (aspect-oriented) Software Reuse
  - Q7 in the OpenOME